UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE ENGENHARIA DE COMPUTAÇÃO

DANIEL SOUZA

End-to-end Bone Age Assessment with Residual Learning

Work presented in partial fulfillment
of the requirements for the degree of
Bachelor in Computer Engeneering

Advisor: Prof. Dr. Manuel Menezes de Oliveira Neto
Coadvisor: Prof. Dr. Fabien Soulier

Porto Alegre
July 2018

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
Reitor: Prof. Rui Vicente Oppermann
Vice-Reitora: Profª. Jane Fraga Tutikian
Pró-Reitor de Graduação: Prof. Vladimir Pinheiro do Nascimento
Diretora do Instituto de Informática: Profª. Carla Maria Dal Sasso Freitas
Coordenador do Curso de Engenharia de Computação: Prof. Renato Ventura Henriques
Bibliotecária-chefe do Instituto de Informática: Beatriz Regina Bastos Haro

“If you don’t know anything about computers, just remember that they are machines that do exactly what you tell them but often surprise you in the result.”

— Richard Dawkins

ACKNOWLEDGMENTS

Firstly, I would like to express my sincere gratitude to my advisor, Prof. Dr. Menezes de Oliveira Neto, for all the support provided not only in this, but also previous endeavors; for his patience, motivation, knowledge, and enthusiasm that helped me accomplish this task. I would also like to most kindly thank all the support and hard work of Prof. Bernardo Henz, Mr. Álesson Scapinello, and Mr. José Eduardo Venson from Iara Health. Being all part of our staff for the RSNA challenge, this project would surely not have been possible without all their hard work, knowledge and motivation. I would also like to thank Animati - Computação Aplicada à Saúde, and Medvia Diagnóstico, for both companies greatly helped me in the pursue of this project, providing a space where this research could be developed. I would like to thank all the support of the radiological staff of Medvia, whose expertise was essential for this project. I would like to thank all of those who helped me on this project in a non-technical way. To all my family and friends, your support, care-giving, and patience were indispensable to attain the expected results, and I cannot thank you enough.

ABSTRACT

Bone age is a reliable metric for determining the level of biological maturity of children and adolescents. Its assessment is a crucial part of the diagnosis of a variety of pediatric syndromes that affect growth, such as endocrine disorders. The most commonly used method for bone age assessment (BAA) is still based on the comparison of the patient's hand and wrist radiograph to a bone age atlas. Such a method, however, takes considerable time, requires an expert rater, and suffers from high inter-rater variability, thus impacting a precise diagnostic.

We present a deep-learning-based approach to estimate bone age from radiographs. It provides a fast, deterministic solution for bone age assessment. We demonstrate the effectiveness of our method by using it to rate a set of 200 radiographs and comparing the results against ground truth. This experiment has shown that our method's performance is similar to the one of a trained physician. Our system is available on-line, providing a free global service for doctors working in remote areas or in institutions with no specialized radiologists as well as for the general population.

Keywords: Machine learning. Convolutional neural networks. Bone age. Radiology.

Avaliação de Idade Óssea Utilizando Aprendizado Residual

RESUMO

A idade óssea é uma métrica usada para estimar o nível de maturidade biológica de crianças e adolescentes. Sua avaliação é parte crucial do diagnóstico de uma variedade de enfermidades pediátricas que afetam o crescimento, como distúrbios endócrinos. O método mais comumente utilizado para a avaliação da idade óssea ainda é baseado na comparação da radiografia de mão e punho do paciente com um atlas de idade óssea. Este método, entretanto, requer um tempo considerável, necessita de um avaliador experiente, e sofre de alta variabilidade entre avaliadores, prejudicando um diagnóstico preciso.
Nós apresentamos uma abordagem baseada em *deep learning* para estimar a idade óssea a partir de radiografias. Nossa abordagem proporciona uma solução rápida e determinística para avaliação de idade óssea. Nós demonstramos a efetividade do nosso método usando o mesmo para avaliar um conjunto de 200 radiografias, comparando os resultados com avalições feitas por radiologistas. Este experimento mostrou que a perfomance do nosso método é comparável à perfomance de um radiologista experiente. Nosso sistema está disponível *on-line*, proporcionando um serviço global e gratuito para médicos trabalhando em áreas remotas ou instituições sem radiologistas especializados, assim como para a população em geral.

Palavras-chave: Aprendizado de máquina, Redes neurais convolucionais, Idade óssea, Radiologia.

LIST OF ABBREVIATIONS AND ACRONYMS

ANN	Artificial Neural Network
BAA	Bone Age Assessment
CNN	Convolutional Neural Network
CR	Computed Radiography
CT	Computed Tomography
CCC	Concordance Correlation Coefficient
DR	Digital Radiography
GPU	Graphics Processing Unit
Grad-CAM	Gradient-weighted Class Activation Mapping
JSON	JavaScript Object Notation
MAE	Mean Absolute Error
MR	Magnetic Resonance
MSCOCO	Microsoft COCO: Common Objects in Context
MSE	Mean Squared Error
PACS	Picture Archiving and Communication System
PNG	Portable Network Graphics
RSNA	Radiological Society of North America
ROI	Region Of Interest
SGD	Stochastic Gradient Descent

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

CONTENTS

1 INTRODUCTION

A child's bone age, or skeletal age, is an indicator of her/his level of biological and structural maturity, which may not agree with the child's chronological age. Both delayed (SALAM, 2014) and increased bone age (SHETTY; SALAM, 2014) can be symptoms of more serious pediatric disorders, hence the importance of bone age assessment (BAA). BAA is also used to estimate the chronological age of children when accurate birth records are not available (MUGHAL; HASSAN; AHMED, 2014).

However, making good estimates of skeletal maturity is a complex and specialized task, requiring detailed examination of many related factors and an understanding of the processes associated with bone development (GILSANZ; RATIB, 2005). BAA methods vary based on the skeletal element being visualized, visualization technique, and may even provide different estimates. Usually, bone age is assessed by visualizing hand and wrist bones, though it is possible – at a certain level – either by visualizing dental maturity, clavicle bone, iliac bone, or femoral head (MUGHAL; HASSAN; AHMED, 2014). The most common visualization method is the visualization of radiographs (see Section 4.1), for its simplicity and history of clinical usage. Other visualization methods include visualization of ultrasound, Computed Tomography (CT), and Magnetic Resonance (MR) images, but they are either slower or not as accurate as the visualization of radiographs. When visualizing radiographs of the hand and wrist, simply put, one has to analyze the growth and deposition of calcium in regions undergoing ossification (GILSANZ; RATIB, 2005).

In Figure 1.1, we can see the phalanges, as well as the metacarpal and carpal bones, used as reference for BAA in different stages of skeletal maturity. The colored regions in this figure are epiphyses, the final portions of long bones. A diaphysis is the mid-section of the bone, while a metaphysis is the narrow portion of the bone between the epiphysis and the diaphysis. By convention, the left hand is used for BAA.

Currently, the most common ways of performing BAA are the Greulich-Pyle (GP) (GREULICH; PYLE, 1959) and Tanner-Whitehouse (TW2) methods (TANNER et al., 1975). GP is an atlas-based solution, meaning that bone age is estimated by comparing the patient's radiograph with the most similar standard radiograph on a gender-specific atlas, as seen on Figure 1.2. Today, digital atlases, such as Gilsanz and Ratib (2005) and Gaskin et al. (2011), make the evaluation process more convenient, but the actual assessment still depends on the rater's expertise (BUNCH et al., 2017).

Figure 1.1: Diagram of hand and wrist bones. Source: Adapted from (ACADEMY, 2016).

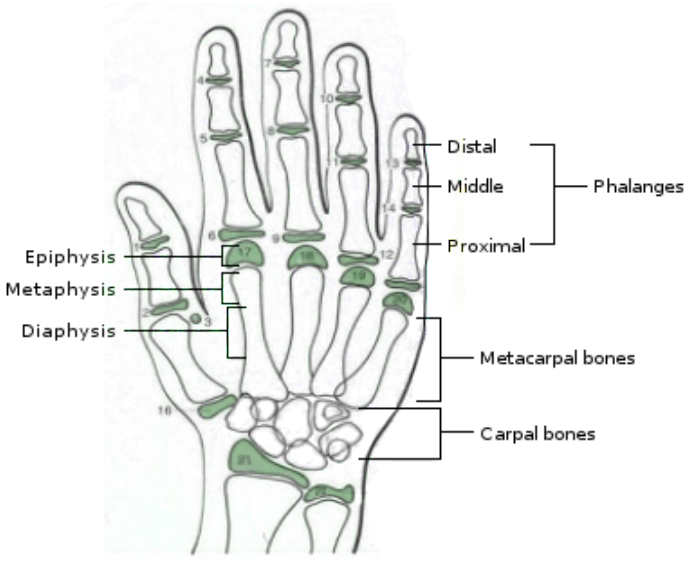


Figure 1.2: A sample radiograph of a seven-year-old boy on a digital atlas. Source: (OESTREICH, 2005).

TW2 is an improvement on the original Tanner-Whitehouse (TW1) method (TANNER; WHITEHOUSE, 1959), and consists of analyzing twenty Regions Of Interest (ROIs) in the hand and wrist and assigning a discrete stage of skeletal maturity (*e.g.*, pre-puberty, early-and-mid puberty, late-puberty, post-puberty) to each ROI. Each stage has an associated score, and a table is used to convert the sum of all scores into a bone age estimate. Since TW2 uses a scoring method, it is normally regarded as being more objective than the GP or other atlas-based methods (SATOH, 2015).

There is an inherent uncertainty in the estimates obtained with both methods. The GP standard radiographs are from a 1931-1942 study conducted with white upper-middle class American boys and girls, and does not take into account ethnic variability. Both GP and TW2 are time-consuming methods, require expert raters, and suffer from high inter-observer variation – disparity between assessments made by different observers for the same radiograph – and intra-observer variation – disparity between assessments made by

the same observer for the same radiograph. Such variability can be critical when making decisions about the most appropriate therapy to be used for each case (LEE et al., 2017).

BAA stands as a natural application for machine learning techniques. Deep *Convolutional Neural Networks* (CNNs) have achieved or even surpassed human performance on several image-related tasks, such as classification (RUSSAKOVSKY et al., 2015), and detection (LIN et al., 2014) of daily objects. They also have quickly become the state-of-the-art for several medical image analysis tasks (LITJENS et al., 2017), including classification (ESTEVA; KUPREL; NOVOA, 2017), segmentation (RONNEBERGER; FISCHER; BROX, 2015), and image generation and enhancement (LI; ZHANG; SUK, 2014).

We present an automated approach for BAA based on CNNs. We train and validate the performance of our solution on a partially-public dataset containing over 12,500 radiographs from the Pediatric Bone Age Challenge (RSNA, 2018), a competition for automating BAA, organized by the Radiological Society of North America (RSNA). We propose a deep, end-to-end solution using *residual learning*. We apply pre-processing stages that are responsible for resizing, standardizing and performing feature scaling on the input data before training our network. To help doctors understand the decision process that led our CNN to its results, we overlay colored *gradient-weighted activation maps* (Selvaraju et al., 2016) (Grad-CAMs) on the evaluated radiographs, highlighting the most relevant structures for each assessment. The accuracy obtained with our solution is comparable to the one of a trained radiologist, and superior or equal to state-of-the-art automated methods, while being easier to apply to different datasets and requiring less dataset preparation and training time when compared to other deep-learning-based solutions. Our system is available on-line, providing a free global service that is particularly relevant for doctors working in remote areas or in institutions with no BAA experts.

The **contributions** of our work include:

- An end-to-end solution for BAA whose accuracy is similar to the one of expert radiologists (see Chapter 5). Our approach uses a CNN with residual learning, and can be easily extended with new data;

- An analysis of the most important hand and wrist structures for bone age assessment performed by a CNN that handles BAA as a regression task (Chapter 6). The results of the analysis are presented for each evaluated radiograph as overlayed color maps indicating the weight of each structure for the BAA;

- A free global on-line BAA service for doctors working in remote areas or in institutions with no specialized radiologists.

We divided this paper in the following chapters:

- Related Work: In this chapter, we present and compare the existing solutions for automated BAA;
- Dataset and RSNA Challenge: In this chapter, we present the RSNA challenge and the dataset provided, while analyzing the dataset and introducing relevant concepts;
- Theoretical Background: In this chapter, we introduce the theoretical machine learning and radiological concepts necessary to understand our approach;
- Our Approach on Automated Bone Age Assessment: In this chapter, we present our solution to automate BAA and the results obtained;
- Discussion and Visualization of Relevant Features: In this chapter, we discuss our results by introducing visualization techniques;
- Conclusion and Future Work: In this chapter, we summarize the contributions of our work and propose future work.

2 RELATED WORK

Several researchers have attempted to develop automatic image processing solutions for estimating skeletal maturity (ZHANG; GERTYCH; LIU, 2007; THODBERG et al., 2009; SOMKANTHA; THEERA-UMPON; AUEPHANWIRIYAKUL, 2011; SEOK et al., 2012). These techniques try to detect and measure features from the radiographs, but were unable to handle the high-variability observed in the development of the hand and wrist bones.

BoneXpert (THODBERG et al., 2009) uses image processing algorithms to automate BAA, and has been approved for clinical use in Europe. It segments 15 bones in hand and wrist radiographs and uses the extracted shape, intensity, and textural features to infer bone age using either GP or TW2 method. The BoneXpert system was developed and validated on small datasets – developed with 1559 radiographs and validated on the GP atlas and 84 additional clinical studies assessed using the TW2 method. Each study has, in general, a single radiograph. It rejects the input radiograph if it fails to detect more than 8 individual bones, if the deviation in the asssessed bone age for indivual bones is too elevated, or even radiographs it deems too noisy. According to a previous study (ZHANG; LIN; DING, 2016), BoneXpert has rejected around 4.5% of individual bones from 397 patients. BoneXpert does not take carpal bones into account, which may negatively impact the BAA for young patients, where these bones have distinguished features.

Somkantha et al. (SOMKANTHA; THEERA-UMPON; AUEPHANWIRIYAKUL, 2011) detect boundaries of carpal bones and extract 5 features from them. These features are used for regression using a support vector machine (SVM). However, they use a small dataset consisting of 180 images of carpal bones extracted from a digital hand atlas. The used radiographs only cover children from 0 to 6 years old, which is a major limitation to their work.

Recently, an automated system for BAA using deep learning was proposed by Lee et al. (LEE et al., 2017). Their approach consists of fine-tuning GoogLeNet (SZEGEDY et al., 2015a) pre-trained on the ImageNet (RUSSAKOVSKY et al., 2015) dataset. They use a pipeline to segment ROIs, standardize, and pre-process input radiographs. Unlike previous approaches, this one uses a bigger dataset that contains 8,325 images. Their technique casts BAA as a classification task, thus rounding all bone ages in their dataset down. This limits their assessments to a 1-year granularity. Their approach achieves a 57.32% and 61.40% accuracy for female and male patients, respectively. Since this is a

classification task, we were unable to properly compare our results to theirs.

Iglovikov et al. proposed an automated system using deep learning developed concurrently with ours for the RSNA challenge, and used the same dataset as we did. Their report is available on arXiv.org (IGLOVIKOV et al., 2017). The authors achieved a Mean Absolute Error (MAE) of 4.97 months on the test dataset of the challenge. Although their solution has a good performance, it is not an end-to-end system, as it performs several pre-processing steps using additional CCNs and required manual intervention at some point of the training process.

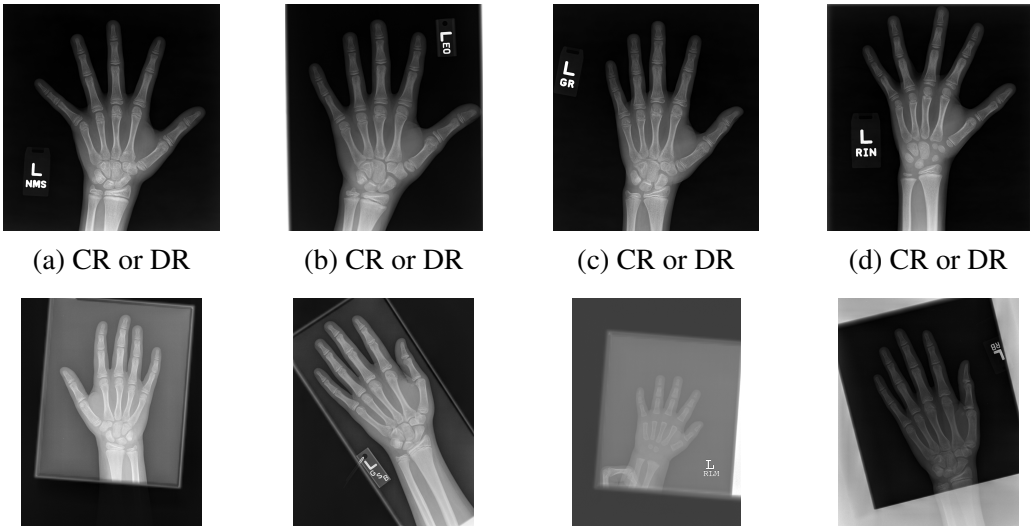
Firstly, they segment the hand and wrist from the input radiographs using a U-Net (RONNEBERGER; FISCHER; BROX, 2015), a CNN architecture originally conceived for biomedical image segmentation. Using such U-Net to segment the radiographs involved generating a training dataset, which required manual generation of segmentation masks, even though it is possible to automate this process to a certain level. They also translate and rotate the radiographs so that the hand bones have a desired position and orientation. This is done by identifying key points on the hand bones. In order to identify these key points, they use an additional CNN based on the VGG architecture (SIMONYAN; ZISSERMAN, 2014a) with a regression output. This CNN also requires manual label generation. Training this solution on new data that differs largely from the original one would probably require manual label generation for all these pre-processing steps. One example of such data would be radiographs that show both the left and right hand and wrist, which has never appeared on any of the challenge datasets. Though not clinically ideal, such radiographs may be frequent depending on the exam's source.

3 DATASET AND RSNA CHALLENGE

We used the dataset of hand and wrist radiographs in PNG (Portable Network Graphics) format with their respective bone ages provided by the RSNA as part of the Pediatric Bone Age Challenge. Teams from all around the world have participated in this competition. This competition had three phases: Training (started on 05/08/2017), Leaderboard (started on 01/09/2017), and Test (started on 07/10/2017, and ended on 17/10/2017). In the Training phase, competitors were given access to the training dataset, which consisted of over 12,500 radiographs with the corresponding bone age and gender of each patient. In this phase, competitors were able to start developing their solutions. Once the solution was ready, it was possible to submit the resulting validation MAE to the challenge's on-line result submission system. For all phases, competitors were able to submit their results to this system and compare them to those obtained by other competitors. For this phase, the competitor could submit its results to a maximum of 20 times. With the start of the Leaderboard phase, RSNA provided a new dataset for testing, consisting of over 1,400 radiographs with the corresponding gender of each patient. Competitors were able to perform BAA for this dataset and submit the assessed bone ages to the result submission system. The competitor was then informed the MAE and Concordance Correlation Coefficient (CCC) for the given bone ages, and the submission was added to the leaderboard. For this phase, the competitor could submit its results to a maximum of 3 times. By performing the BAA for this dataset, competitors were able to have a finer idea of how their approach would work on unseen data, and consequently, the final Test phase. In this phase, competitors were given access to a dataset consisting of 200 radiographs with the corresponding gender for each patient. Competitors were able to perform the BAA for the test dataset and submit the assessed bone ages to the result submission system. The competitor was then informed the MAE and CCC for the given bone ages, and the submission was ranked with other submissions. For this phase, the competitor could submit its results to a maximum of 3 times. By the end of this phase, the competitor with the lowest MAE would win the challenge, being the CCC used as a tie-breaker.

The dataset had contributions from the Stanford University, the University of Colorado, and the University of California - Los Angeles. This dataset presents a high variability when it comes to the aspect of the radiographs, being the different methods used to acquire these images as well as the different sources important causes. The radiographs may vary in brightness, contrast, resolution, and even aspect ratio. In Figure 3.1 we see 8

Figure 3.1: 8 sample radiographs available on the training dataset. Source: The authors.



(a) CR or DR (b) CR or DR (c) CR or DR (d) CR or DR

(e) Digitized film (f) Digitized film (g) Digitized film (h) Digitized film

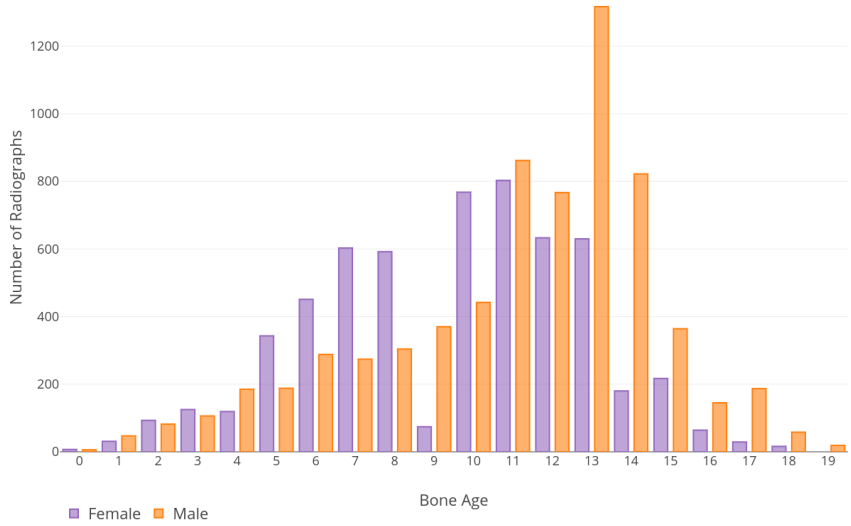
radiographs (see Section 4.1) extracted from the training dataset, which contains images acquired trough processes such as Computed Radiography (CR) or Digital Radiography (DR) – which usually result in a clean, proper image – and images that were digitized from traditional film – which may not always result in a proper image. Though experienced radiologists may not struggle with assessing the correct bone age of such cases, we believe a neural network would benefit from a cleaner, more standardized input.

Since BAA is performed to identify growth disorders, there is a natural unbalance to the age distribution of patients, which is also seen on the challenge's dataset. In Figure 3.2, we can see that the dataset's age distribution is unbalanced for both genders, and that these distributions are largely different for each gender. Ideally, datasets used for training machine learning solutions should be balanced. Therefore, we believe that this may negatively impact the performance of our algorithm for the ages with the least studies available.

In accordance with the challenge's rules, we used the MAE, and CCC metrics to measure our performance. Given that the prediction error can be defined as the difference between the ground-truth value and the predicted one, the MAE of a model is the mean of the prediction errors for all instances in the test set (SAMMUT; WEBB, 2010). The CCC is a coefficient used to measure the agreement between two continuous variables, in our case, the predicted bone age and the ground truth.

Given a sample i, the ground truth for that sample x_i, and the predicted bone age

Figure 3.2: Age Distribution in the RSNA Training Dataset. Source: The authors.



for that sample y_i, the MAE is obtained by:

$$MAE = \frac{\sum_{i=1}^{n} |x_i - y_i|}{n} \tag{3.1}$$

Given the means for, respectively, the ground truth and the predicted bone age $\bar{x}$ and $\bar{y}$, the corresponding variances s_x^2 and s_y^2, and the covariance s_{xy}, we define $\hat{\rho}_c$, the CCC as:

$$\hat{\rho}_c = \frac{2s_{xy}}{s_x^2 + s_y^2 + (\bar{x} - \bar{y})^2} \tag{3.2}$$

Where the means can be calculated given the equations, assuming N is the size of our test dataset:

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i \quad \text{and} \quad \bar{y} = \frac{1}{N}\sum_{i=1}^{N} y_i \tag{3.3}$$

and the corresponding variances:

$$s_x^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^2 \quad \text{and} \quad s_y^2 = \frac{1}{N}\sum_{i=1}^{N}(y_i - \bar{y})^2 \tag{3.4}$$

and the covariance:

$$s_{xy} = \frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})(y_i - \bar{y}). \tag{3.5}$$

4 THEORETICAL BACKGROUND

4.1 Radiography

A radiograph can be defined as an image produced by radiation. While other imaging techniques may be out of the scope of this project, it is important to have a clear understanding of how a radiograph is obtained, for it is the type of image in the challenge's dataset and the most used visualization method for BAA. Radiographs may be obtained by screen-film radiography, or newer methods such as Computed Radiography (CR) and Digital Radiography (DR).

CR is the use of photostimulable phosphor as an image receptor. The process is straightforward (BELL; MURPHY, 2017):

- Firstly, the image receptor plate is exposed to X-ray or gamma radiation, storing the image.
- A focused laser releases the stored image in the form of visible light.
- Finally, the emitted light is then read by a scanner, converted to a digital signal and transferred to a computer.

DR, a process similar to CR, uses digital radiation sensors instead of photostimulable phosphor (KöRNER et al., 2007). While CR does not use digital sensors, both DR and CR produce digital images.

Screen-film radiography can also be used to obtain radiographs, but it does not produce digital images, requiring the use of a scanner or other digitization techniques.

4.2 General machine learning concepts

A machine learning algorithm is an algorithm that has the ability to learn from given data. According to Mitchell (1997) "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E". A learning task can be defined as the manner a machine learning algorithm should process an input vector in order to attain an objective. In other words, a general learning task can be seen as a credit assignment problem (MINSKY, 1961). Simply put, the credit assignment problem consists in determining how the various contributions of a system's components impact

the system's performance. The act of experiencing a dataset – a collection of samples – or part of it determines many important characteristics of machine learning algorithms. These algorithms can be classified as *unsupervised*, *supervised*, or *reinforcement* learning algorithms, depending on their experience E. We will be focusing mostly on the difference between unsupervised and supervised learning algorithms:

- Unsupervised learning algorithms experience a dataset and try to infer patterns from the structure of this dataset.
- Supervised learning algorithms experience the same feature dataset, but in this case, each sample has an associated label indicating its expected output; In both Sections 5.1 and 5.2, we proposed approaches based on supervised learning algorithms.

We are particularly interested in two classes of supervised learning tasks: regression, and structured output, for they are used for BAA and segmentation, respectively. Finally, given a task, it is important to define a metric to evaluate how well the model performs.

4.2.1 Artificial neural networks

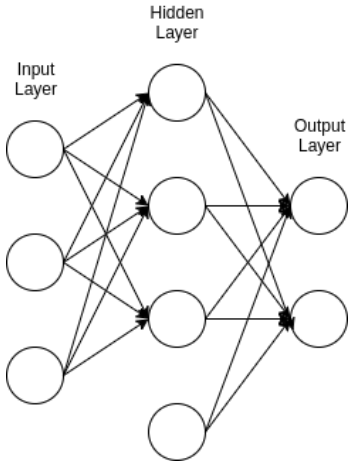
An Artificial Neural Network (ANN) is a computational model widely used in machine learning, and can be defined as a set of many processing units connected to each other. These processing units are called neurons, for they were conceived to mimic the behaviour of biological neurons. Each neuron produces an output, which is a real-valued activation. Input neurons provide data to the neurons it connects to, while other neurons get activated by weighted sum of the outputs of the previously activated neurons connected to them (SCHMIDHUBER, 2014). ANNs may also contain bias units, which feed an input to a following neuron. Unlike regular neurons, bias units do not take into account the output of previously activated neurons.

An ANN can also be seen as a set of many layers, which can be classified as input, hidden, and output layers (see Figure 4.1). Each layer consists of one or more neurons. An input layer is a set of input neurons, while an output layer is a set of the final neurons of the network, in the sense that their output is not used as an input to a following layer. The hidden layers are those in-between the input and output layer.

In neural networks, the credit assignment problem consists of finding a combination of weights for each neuron that makes the neural network behave as desired – *e.g.*,

correctly estimating the bone age of a patient. This happens during a process known as *training*. When an ANN is used to approximate a function, the input vector is forwarded through the network, layer by layer, applying the current weight of each neuron to the output of the previous layer. The training process attempts to minimize a *loss function* by using *gradient descent optimizers*.

Figure 4.1: A simple feed-forward ANN with its layer types. Source: The authors.



4.2.2 Loss function

A loss function can be defined as a function that computes how far an evaluation of the current weights – *e.g.*, the predicted bone age – is from the ground truth. We use the Mean Squared Error (MSE) as our loss function, where our error can be defined as the difference between our predicted and ground truth sets. Given a sample i, the ground truth for that sample x_i, and the predicted bone age for that sample y_i, the MSE is obtained by:

$$MSE = \frac{\sum_{i=1}^{n}(x_i - y_i)^2}{n} \tag{4.1}$$

4.2.3 Gradient Descent Optimizers

Gradient descent is a very popular algorithm for optimization, and is largely used in machine learning to minimize the loss function. The gradient is a multi-variable generalization of the derivative with respect to a vector. The gradient of a given function is a vector field that points towards the greatest rate of increase of that function (GOODFELLOW; BENGIO; COURVILLE, 2016). We may obtain the gradient of our loss function

by computing the partial derivatives of the loss function with respect to all weights and biases with *backpropagation*. Therefore, a gradient descent optimizer updates its parameters – our network's weights – by taking a step in the opposite direction of the gradient of a loss function. The gradient does not, however, inform us how far should we follow the pointed direction, which is why it is important to define the size of this step, the learning rate. Choosing the right value for the learning rate can be a hard, laborious task.

Vanilla gradient descent optimizer updates its parameters after computing the gradient of the loss function for the entire training dataset, which is very slow and requires a lot of memory. The Stochastic Gradient Descent (SGD) solves this problem by updating its parameters after computing the gradient of the loss function for a single sample (RUDER, 2016). Being simple and efficient, if compared to vanilla gradient descent, SGD was the base for faster, more complex optimizers.

Momentum is an SGD based optimizer. In Momentum, an update of parameters will take into account the average of past gradients. It does so by adding a fraction of the last parameter update to the current update. This brings faster convergence and reduces oscillations often seen with SGD.

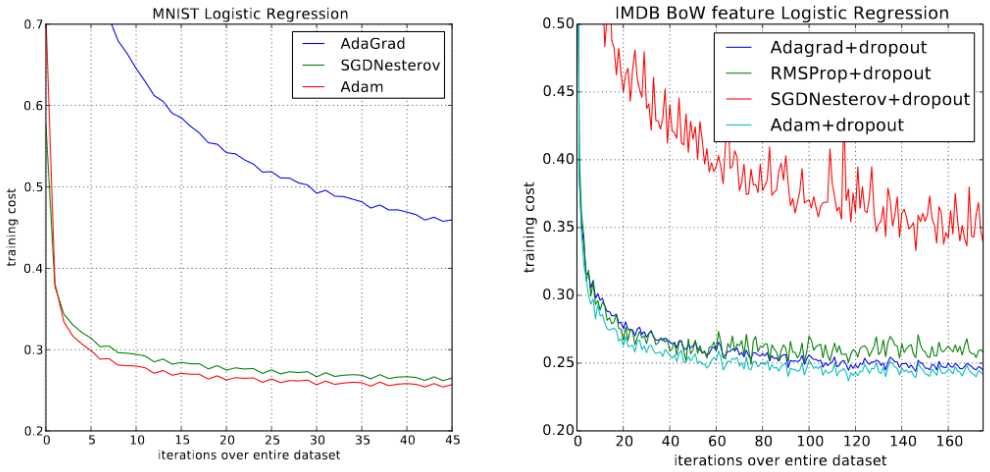
AdaGrad (DUCHI; HAZAN; SINGER, 2011) is another SGD based optimizer. Unlike previous optimizers, AdaGrad is able to adapt the learning rate to the parameters. In AdaGrad, an update of parameters will take into account the sum of squares of past gradients. Given the parameter ω, the learning rate η, the diagonal matrix G_t where each diagonal element is the sum of the squares of the gradients with respect to the current sample up to step t, a smoothing term ϵ – used to avoid divisions by zero – and g_t, the gradient of the loss function at iteration t, AdaGrad's parameter update function can be defined as:

$$\omega_{t+1} = \omega_t - \frac{\eta}{\sqrt{G_t + \epsilon}} \times g_t \tag{4.2}$$

This leads to parameters associated with infrequent features causing bigger updates (big learning rate), while parameters associated with frequent features cause smaller updates (small learning rate). Still, AdaGrad has a major problem: since the sum of squares of the past gradients is always positive, this sum is always growing. It is easily observable from Equation 4.2 that such behavior would cause the parameter change to become smaller and smaller, to the point where the parameter update becomes irrelevant and we are unable to learn new features.

RMSProp (TIELEMAN; HINTON, 2012) is based on AdaGrad, and aims to solve

Figure 4.2: The training loss of a model using various optimizers. Each plot shows a different model and dataset. Source: (KINGMA; BA, 2014).



the diminishing parameter updates problem. The main difference between their parameter update functions is that RMSProp uses a decaying average of past squared gradients instead of a simple sum, solving the proposed problem.

The Adaptive Moment Estimation optimizer, more commonly called Adam optimizer (KINGMA; BA, 2014), like AdaGrad, is able to adapt the learning rate to the parameters. Adam takes the best of RMSProp and Momentum, meaning that it stores an exponentially decaying average of past squared gradients – taken from RMSProp – and an exponentially decaying average of past gradients – taken from momentum.

In Figure 4.2, we can see a comparison of the behavior of a model's loss when different optimizers are used. It becomes clear that, by using Adam, we are more likely to achieve better results. The figure does not, however, compare the loss when standard SGD or vanilla gradient descent optimizers are used, for they are far worse than the optimizers shown, and may even fail to converge to a good local minima.

4.2.4 Convolutional neural networks

Convolutional Neural Networks (CNNs) are a class of ANNs that can be defined as a set of convolutional layers optionally followed by a set of fully-connected layers (layers where every neuron is connected to every neuron in the following layer). Any layer that has neurons that perform convolutions is called a convolutional layer. CNNs are also feed-forward, meaning that there are no feedback connections – the output of a neuron can only be fed to a neuron in a following layer, not fed back to a previous

layer (GOODFELLOW; BENGIO; COURVILLE, 2016).

In the CNN context, the convolution operation may differ slightly, in practice, from the mathematical definition of a discrete convolution (GOODFELLOW; BENGIO; COURVILLE, 2016). Considering our input is an image, our convolutional layers perform the convolution of a kernel, or filter, over an image to produce a feature map, or activation map.

CNNs are very adequate for handling image-related problems, for they have three important notions (GOODFELLOW; BENGIO; COURVILLE, 2016):

- Sparse interactions: Each neuron on a convolutional layer only feeds a small number of neurons on the next layer. This is achieved by making the kernel size smaller than the input image;
- Parameter sharing: A kernel is reused in different positions of the input image. This allows the network to learn spatial structures;
- Equivariant representations: A deviation in the input will cause the output to change in the same way. This helps our network to generalize patterns in different locations;

These three notions allow CNNs to learn and generalize patterns such as edges, while dramatically improving memory usage and performance when handling images, if compared to regular ANNs.

Typically, a convolutional layer is composed of three stages: the *convolution stage*, the *detector stage*, and *pooling stage*. The convolution stage performs the convolution of a kernel over the input, producing an ensemble of linear activations. The detector stage passes each linear activation to a non-linear activation function, which allows CNNs to discover more complex patterns. The pooling stage may be used to down-sample the feature maps provided by the detector stage (GOODFELLOW; BENGIO; COURVILLE, 2016). Therefore, the pooling stage also produces feature maps, though each unit in its feature map is computed based on a subset of the units from the detector stage's feature map. This stage reduces the size of feature maps, and therefore, the amount of parameters to be learnt. It boosts translation invariance, meaning that small spatial shifts in the input may produce the same activation map after passing through the pooling stage (ZHOU; GREENSPAN; SHEN, 2017). A commonly used pooling method, is the max pooling (ZHOU; CHELLAPPA, 1988), which reduces the dimension of the input feature map by only keeping the maximum value within a rectangular range.

4.2.5 Normalization

Normalizing the input data is a crucial step for training neural networks. There are several normalization techniques, from which we used *feature scaling* and *standardization*.

Feature scaling consists in normalizing the range of independent variables in a dataset. For both train and test datasets, we apply feature scaling. We rescale the values of the pixels – 8-bit unsigned integers – of each image to fit the [0,1] range, given the formula:

$$x_{norm} = \frac{x - min(x)}{max(x) - min(x)} \tag{4.3}$$

Where x is the value for a given pixel and x_{norm} is the normalized value for that same pixel, and $min(x)$ and $max(x)$ are the minimum and maximum value for a pixel in that image, respectively.

Standardization, also commonly referred to as z-normalization, or normalization to zero mean and unit variance, consists in transforming a dataset to have its mean approximately equal to zero, and its variance approximately equal to one. Given a value from an input vector x, the mean of this vector μ, and its standard deviation σ, the normalized output vector x_{norm} is given by:

$$x_{norm} = \frac{x - \mu}{\sigma} \tag{4.4}$$

For both train and test datasets, we apply standardization. However, the test dataset is standardized using the mean and standard deviation of the training dataset. This is optimal, since the test dataset is small, meaning that its mean and standard deviation are more prone to sampling errors than those of the training dataset.

One can easily understand the impact of normalizing the input data by looking at Figure 4.3. This figure shows some arbitrary data, its standardized representation, and the feature-scaled representation of its standardized version.

Figure 4.4a shows an example of a radiograph used in our test dataset, while Figure 4.4b shows a brighter version of that radiograph. While visually different, our neural network should evaluate the same bone age for both radiographs, which is ensured by the use of normalization.

Figure 4.3: Different normalization techniques applied to arbitrary data. Source: Adapted from (RASCHKA, 2014).

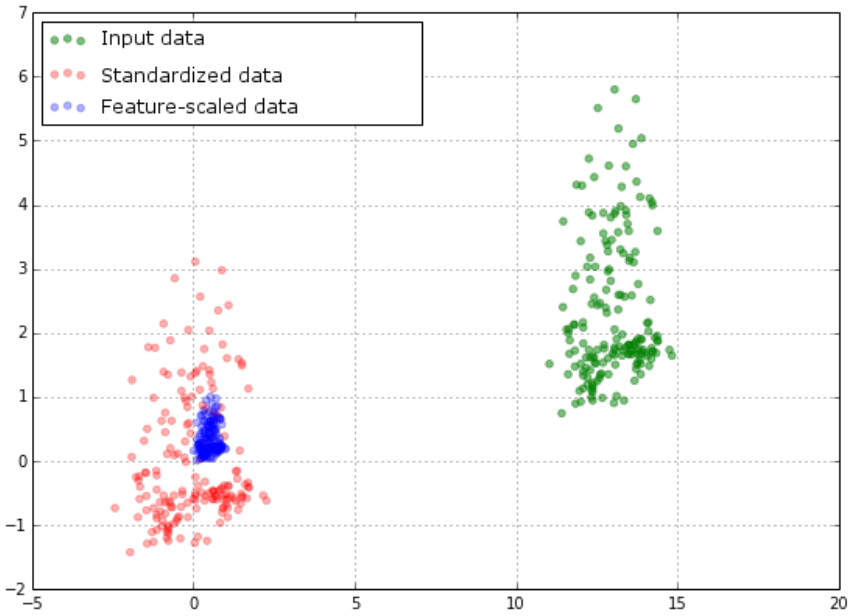
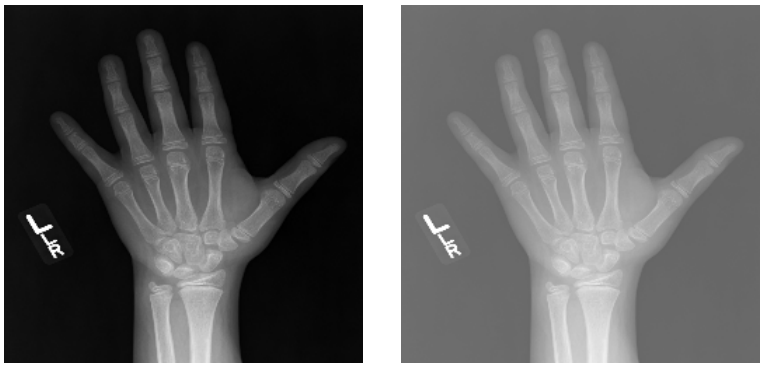


Figure 4.4: Two images with different brightness. Source: The authors.



(a) Original image (b) Same image, but brighter

4.2.5.1 Batch Normalization

Much like we standardize the inputs to our network, we also standardize the inputs to each hidden layer. Batch normalization (IOFFE; SZEGEDY, 2015) manages to standardize the inputs to the hidden layers by subtracting the batch's mean, and then dividing by the batch's standard deviation. This process has several benefits, such as decreasing convergence time and adding some regularization to our network.

4.2.5.2 Instance Normalization

Instance normalization (ULYANOV; VEDALDI; LEMPITSKY, 2016) is another normalization method, more recent than batch normalization. For now, it was mostly used

with Generative Adversarial Nets (GOODFELLOW et al., 2014) based architectures, such as Cycle-GAN (ZHU et al., 2017). Instead of performing normalization within a batch, instance normalization does it within a single sample.

4.2.6 Overfitting

One major problem encountered when training a neural network is overfitting. Overfitting means that our neural network has learned not only the relevant information from our training dataset, but also its noise. Learning the noise of a dataset means that small specific features are learned as features that could be generalized to any data, which is not true. In other words, our neural network modeled our training data so well that it negatively impacts our test accuracy. In order to prevent overfitting, we added regularization techniques (see Section 4.2.8) .

4.2.7 Underfitting

Another major problem encountered when training a neural network is underfitting. One may think of underfitting as the opposite of the overfitting problem. When our neural network underfits our data, it means that it can neither model the training data nor generalize new data – i.e. our test data. It is normally an indicator that our model is far too simple, which is why we make our neural networks progressively deeper and add nonlinear activation functions to it (see Section 5.2).

4.2.8 Regularization

Regularization consists in trading-off flexibility and model complexity as to avoid overfitting, for instance. We will be using two regularization methods:

- Dropout: In the fully connected layers, neighboring neurons often develop a co-dependency. This means that if our network were to evaluate the output of two neighboring neurons, one with a large weight and the other with a small weight, it may come to a point were the output of the smaller weight neuron is never considered. Once our network reaches such state, it fails to learn on new data, and

becomes more and more specific to previous seen data – resulting in overfitting. Dropout consists in randomly disconsidering the output of neurons in our network. This gives neurons who were dependent on their neighbors the ability to learn new features and helps generalization.

- L2 regularization: L2 regularization consists in applying a penalty that is progressively higher as our CNN gets deeper. The reason behind this becomes clearer when one thinks of the traditional problem of finding the curve that best fits a given set of points. As the degree of our polynomial grows, we start modeling not only the data, but its noise as well. This causes overfitting, as we would fail to fit new points. In this case, the degree of our polynomial may be compared to the depth of our CNN, as the top-most convolutional layers learn more complex features, which may not generalize well and should have a penalty applied to their weights.

4.2.9 Data augmentation

One way to virtually increase the size of our dataset and, most importantly, make our model more robust to the variability in our data, is using data augmentation. Data augmentation consists in randomly applying transformations to the training data so that it may be used as an entirely new radiograph to train. This makes our model more robust for our network learns that these small deviations in the images – the noise – such as rotations, and zooms, should have no impact on the assessment of the bone age. It is very important to choose transformations that represent data actually present in the dataset. For instance, there are no upside-down images in our dataset, so a vertical flip would not be an appropriate transformation. It is also important to choose the range of these transformations – where it applies – in a way that it actually represents the data. For example, there are no landscape oriented radiographs in our dataset, so it would be inappropriate to use a rotation range that is, in module, as big as $90°$.

Having this in mind, we analyzed our dataset and decided to use the following data augmentation techniques (as seen on Table 5.2):

- Rotation: rotates the radiographs by any random real number ranging from -10 to +10 degrees;
- Horizontal jitter: shifts the radiograph content horizontally by up to 5% of its width;
- Vertical jitter: shifts the radiograph content vertically by up to 5% of its height;

- Zoom: zooms in or out the radiograph by a factor up to 5% of its original size;
- Horizontal flip: This technique consists in randomly applying an horizontal flip to the radiograph. A standard bone age radiograph shows the left hand and wrist. Though most cases were standard radiographs, in some cases we were presented the right hand and wrist. Therefore, this is a very useful augmentation – even if it should not be applied very often.

5 OUR APPROACH ON AUTOMATED BONE AGE ASSESSMENT

Automated BAA comes as a solution to the high intra-observer and inter-observer variability in traditional methods. It can be used to increase productivity of radiologists, and help inexperienced radiologists by fetching similar cases to the radiograph being examined.

Firstly, we proposed our BAA system to be divided in 2 modules: A segmentation module and the actual BAA module. The segmentation module would be responsible for pre-processing the radiographs by segmenting the hand and wrist. This module, however, was able to accurately perform segmentation for a small number of cases only, which we believe is due to lack of training data. Since the process of manually generating segmentation masks for training takes a considerable amount of time, we decided to use the BAA module only. This was necessary to keep ourselves on the challenge's schedule. Due to remarkable performance of deep CNNs, we conceived our BAA module to consist of an end-to-end deep CNN, meaning that the entire BAA task is handled by our CNN, and there are no decapsulated additional steps.

5.1 Segmentation module

Radiographs from the RSNA dataset may include a large amount of information that is irrelevant for estimating the bone age of a patient, such as markers, or bounding rectangles. By performing a semantic segmentation, we aimed to remove all non-hand and wrist elements from the radiograph. This task can be classified as a structured output task, for its output is a data structure where important relations exist between different elements.

We used Facebook Research's DeepMask (PINHEIRO; COLLOBERT; DOLLAR, 2015) and SharpMask (PINHEIRO et al., 2016) to perform semantic segmentation. The code for both projects is available in a public repository (FACEBOOK, 2016) under the BSD 3-clause license (INITIATIVE, 2012), making it simple to reuse. It is implemented in Torch7 (COLLOBERT; KAVUKCUOGLU; FARABET, 2011), a scientific computing framework commonly used for machine learning applications.

Deep Mask uses a feed-forward deep network architecture to perform semantic segmentation. In such architecture, deeper layers have more complex information – semantic features, for example. These same layers, which are used to predict segmentation

masks, are computed at lower spatial resolutions to reduce computational cost and improve translational invariance. This, however, is a problem for accurate segmentation mask generation, which is why SharpMask is also used. SharpMask is responsible for refining the output of DeepMask, by reversing the flow of information of a feed-forward network, therefore using the features learned by progressively earlier layers of DeepMask. These layers learn more basic features, such as edge detection, which is why they are used to refine masks produced by DeepMask.

Both DeepMask and SharpMask expect the ground-truth to follow the Microsoft Common Objects in Context (MSCOCO) dataset annotation pattern (LIN et al., 2014). MSCOCO is a dataset and API ensemble used in several image processing challenges, such as segmentation, and object detection challenges. Its annotation standard for segmentation ground-truth consists of a JavaScript Object Notation (JSON) file, a data-interchange format structured to contain the points, area and bounding box of the segmentation polygon, as well as relevant metadata. We used adaptive thresholding methods, as well as manual segmentation for the cases the first fails, to generate simple binary masks of the hand and wrist. These masks were used to generate more complex MSCOCO standard annotation JSON files, which were used as ground-truth to our training.

To solve the semantic segmentation problem, we used a method known as transfer learning. The principle behind transfer learning is reusing – on new data – generic features learned by a pre-trained neural network, by simply retraining the same network with new data. Transfer learning is particularly useful when there is few data available for training (YOSINSKI et al., 2014).

5.1.1 Implementation

We did not need to make big changes to the DeepMask and SharpMask projects to perform the segmentation. However, building the dataset to train the DeepMask and SharpMask models, as well as applying the segmentation masks to our images required some coding and manual work.

Firstly, we needed to obtain our segmentation ground truth. This task was done both automatically – by using adaptive thresholding –, and manually. For the automatic process, we decided to use OpenCV (BRADSKI, 2000), an open-source library for computer vision. Easy to use, OpenCV has an implementation of the Otsu thresholding (OTSU, 1979), which we chose as our adaptive thresholding technique. The Otsu

thresholding transforms a grayscale image in a binary image by applying every threshold value possible and then calculating the intra-class variance for each class – black or white. Finally, it chooses the threshold that minimises this variance. For the manual process, we decided to use the Fiji (SCHINDELIN et al., 2012) distribution of ImageJ (SCHNEIDER; RASBAND; ELICEIRI, 2012), an open-source image processing program developed by the National Institutes of Health. Fiji contains several plugins that go from medical imaging processing to machine learning. Fiji allows us to manually apply thresholding to generate a ROI. This ROI can then be manually edited to fix finer details where the thresholding technique fails. By using both methods, we were able to generate around 1500 segmentation masks, which were used to create the training dataset.

Having obtained a binary mask trough thresholding or manual editing, one must convert it into a MSCOCO annotation file – the format expected by DeepMask and SharpMask. This format eases the handling of large segmentation datasets, containing relevant metadata, and occupying much less space than an image. Among some metadata – class identifier, image identifier, source, etc – we needed to obtain a polygonal representation to the image of a hand and calculate its area and bounding box. This was easily solved with OpenCV, which provides a method called findContours that can be used to find contours in binary images, returning an array of polygons corresponding to each contour found. By selecting the largest contour, we have the polygonal representation of the hand. OpenCV also provides a method called contourArea, which can be used for calculating the area of a given polygon, and a method called boundingRect, which return the axis-aligned bounding box of a given polygon – which can be obtained from the maximum and minimum values of all vertices along the given axes.

5.1.2 Experiments

To train the DeepMask model, one needs to feed it a pre-trained model used for computer vision tasks, such as classification, so that DeepMask may learn to generate high-level segmentation masks. We used, according to Facebook´s recommendation, the ResNet-50 (HE et al., 2015) model pre-trained on the ImageNet dataset. To train the SharpMask model, a pre-trained DeepMask model is necessary. Also, it is not possible to use transfer learning on the SharpMask model directly. Having this in mind, we proposed three experiments, from A to C:

A. Directly performing the semantic segmentation with the pre-trained SharpMask model provided by Facebook.
B. Training SharpMask by using the pre-trained DeepMask model provided by Facebook, and then performing the semantic segmentation with our SharpMask model.
C. Training a DeepMask model from the ResNet-50 pre-trained model, then training SharpMask with our DeepMask model. Finally, perform the semantic segmentation with our SharpMask model.

5.1.3 Results

We attempted to perform all three proposed experiments, A to C, from which:

- Experiment A showed the worst results. This was expected, since our data is largely different from the data Facebook used to train their models.
- Experiment B could not be performed. In order to use the pre-trained DeepMask model, we would need to use the same network and batch sizes for SharpMask, which we were unable to do due to hardware limitations – specifically, lack of GPU memory.
- Experiment C showed the best results. We believe that by doing so, we were able to leverage the lower-level features learned by the ResNet-50, while being able to learn higher-level features specific to our data by training DeepMask and SharpMask.

The output of the SharpMask network is an RBG segmentation mask, which is largely different from the BAA input. Therefore, we developed a pipeline of 5 steps to perform segmentation and output an image that could be used to perform BAA. The output of each step can be seen at Figure 5.1. The pipeline consists of:

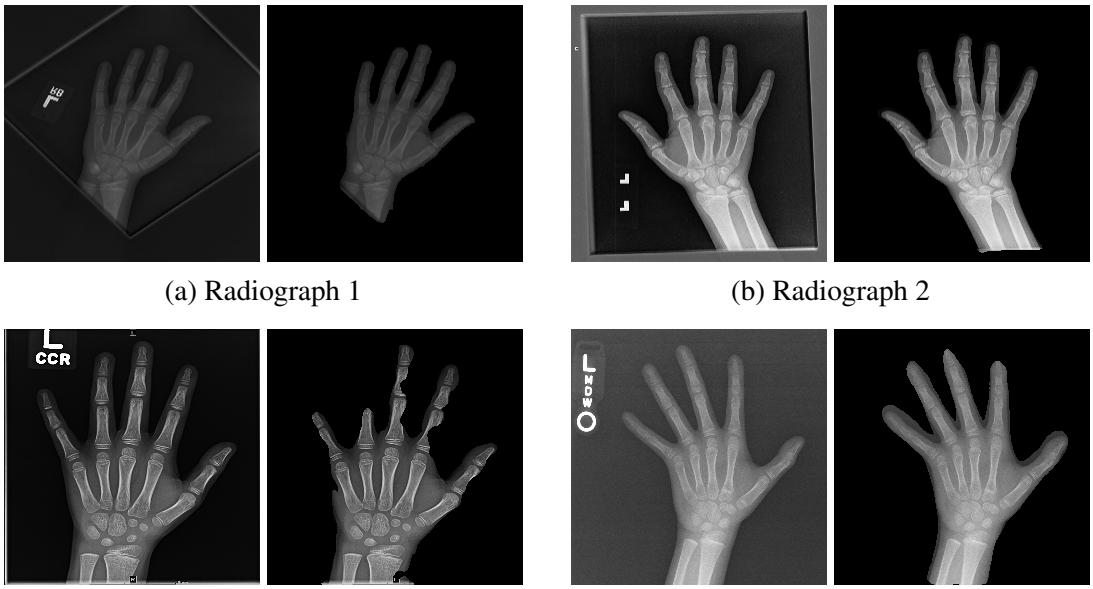
1. Loading the original image.
2. Using our best SharpMask model to generate an RGB segmentation mask.
3. Transforming this RGB mask into a binary mask (background will be black, while the hand will be white).
4. Selecting the largest white region and removing any black regions inside.
5. Using the previous mask to perform a bitwise AND operation on the original image.

Figure 5.1: Our segmentation pipeline. Source: The authors.

(a) Step 1 (b) Step 2 (c) Step 3 (d) Step 4 (e) Step 5

As shown in Figure 5.2, our segmentation pipeline showed fine results for some cases, but it was still not enough to be used for training our BAA module. We trained our model for 20 epochs, where it reached a plateau, probably due to the lack of training images. We believe segmentation might be a candidate for future work (see Chapter 7).

Figure 5.2: 4 radiographs before and after undergoing our segmentation pipeline. Source: The authors.



(a) Radiograph 1 (b) Radiograph 2

(c) Radiograph 3 (d) Radiograph 4

5.2 Bone age assessment module

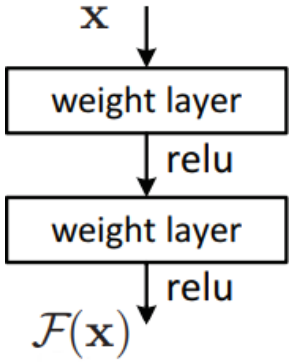
We used deep CNNs to automate BAA, which can be classified as a regression task, meaning that the expected output to a given input is a numerical value, with our loss function being based on the MSE metric. A fundamental part of achieving good performance on a given task is to chose an appropriate architecture and hyper-parameter set. These factors are very experimental and very dependent on specific characteristics of the dataset.

We were particularly interested in comparing the performance of fine-tuning the Inception-V3 architecture (SZEGEDY et al., 2015b) pre-trained on the ImageNet dataset, to training very deep custom CNN architectures. We chose to fine-tune the Inception-V3 network for it has a high performance on the ImageNet dataset, and has several open-source implementations available.

Fine-tuning consists of freezing a number of layers – usually the bottom-most – of a pre-trained network, and retraining the network on your data only. Freezing layers means that these layers will not be retrained – in our case, we freeze the first 5 layers. This technique allows a CNN to benefit from the low-level features extracted from a given dataset, while learning high level features specific to another dataset.

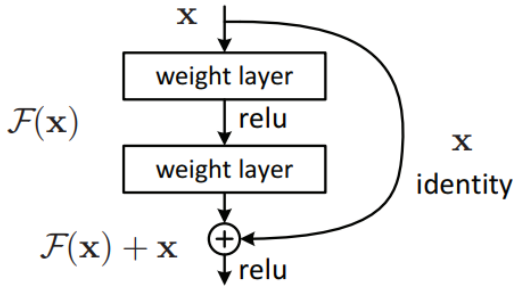
One major problem with training very deep neural networks is that with increasing network depth, the accuracy tends to saturate, degrading quickly. Surprisingly, this is not due to over-fitting, and simply adding layers leads to higher training error (HE et al., 2015). Residual layers solve this problem by using connections called *skip connections*, allowing our network to learn deviations from the identity layer, x. In Figure 5.3, we can see a stack of layers in a traditional neural network, being $F(x)$ the desired mapping. In Figure 5.4, we can see a stack of residual layers – called a *building block*. In such cases, the desired mapping then becomes $F(x) + x$, which is important, because the learning process makes $F(x)$ subject to degradation. Adding the identity to the output is also important in the sense that, in the worst case scenario, our network would perform as well as a shallower version of itself. The original paper hypothesizes that fitting a residual mapping is easier than fitting the desired mapping directly.

Figure 5.3: A plain neural network. Source: Adapted from (HE et al., 2015).



Residual networks – neural networks that use residual layers – can go deeper and learn to represent more complex models without saturating accuracy. These networks attained the 1st-place winning entries in all five main tracks of the ImageNet (RUSSAKOVSKY et al., 2015) and COCO 2015 competitions. For these reasons, we have

Figure 5.4: Definition of a generic building block. Source: (HE et al., 2015).

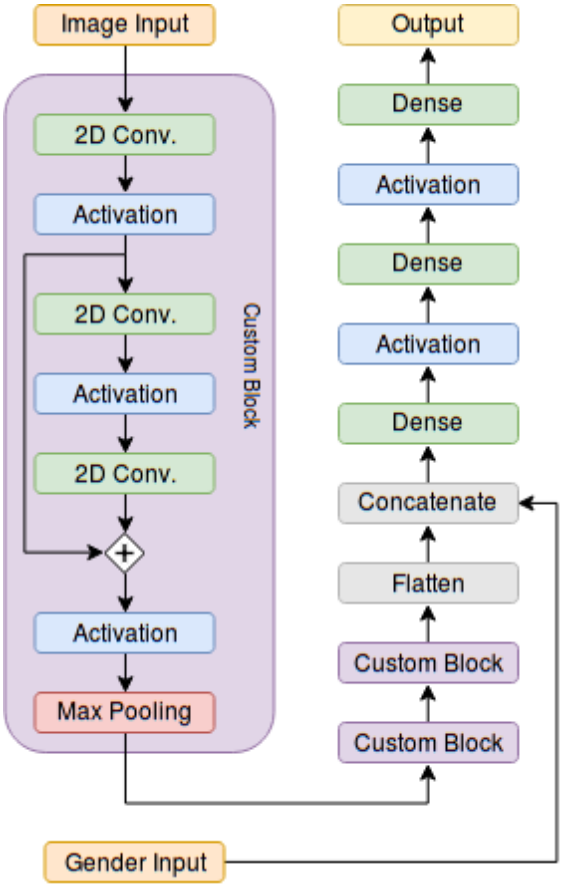


proposed several custom architectures using residual layers that consist, partly, of an ensemble of custom blocks. These custom blocks will be composed of a convolutional layer followed by an activation layer, followed by a number of building blocks ranging from 1 to 3. Each building block has 2 convolutional layers, with an activation in-between them, followed by an addition operator and another activation. Finally, our custom block has a pooling step. Each custom block can be followed by a similar wider block – wider blocks perform convolution on a larger number of filters. By progressively increasing the width of our architecture, we allow our network to learn more from its deeper layers than its shallower layers. Finally, we flatten the output of our custom block ensemble, concatenate it to the patient's gender, and add dense layers with activations in-between to obtain the bone age (see Figure 5.5). Since radiographs of the same age for different genders are largely different, we chose to use it as an input to the dense part – simply put, the non-convolutional part – of our neural network.

5.2.1 Implementation

As previously stated, our main proposal to solve the bone age assessment task is to train a custom network from scratch, but we also aim to experiment with transfer learning techniques with a pre-trained Inception-V3 model. For both proposals, we will be using a neural networks API named Keras (CHOLLET et al., 2015). Keras is a rich, high-level API that allows us to focus on the machine learning concepts rather than the programming itself. Defining complex architectures in Keras is as simple as "stacking" layers, as seen in Algorithm 5.1. Keras is written in Python (DRAKE, 2017) and is capable of running on top of various numerical computation and machine learning libraries such as TensorFlow, CNTK, or Theano. We chose to run Keras on top of Tensorflow (ABADI et al., 2015) for it is largely used by the machine learning community, is open-source, has

Figure 5.5: Representation of candidate architecture A. Source: The authors.



great performance, and is being constantly improved by Google. Tensorflow also provides a visualization tools suite, called Tensorboard, that is easy to use and very useful for debugging and better understanding the outputs of our experiments.

We developed a objected-oriented solution with Python, in a way that we could easily reuse this code for other projects, simply changing parts that are specific to the task at hand, such as network architecture implementation or data pre-processing, loading, and feeding. We defined data, generator, model, test, and train classes. Our data class has the arrays where train, validation and test data will be loaded to, as well as its respective paths and user controlled parameters used to load the data. These parameters are the network input image size, and the percentage of the train dataset to be used as validation. This class also implements methods used to load train and test data, and pre-process the images – normalization, and resizing. Our model class extends the base Keras model class, adding meta-data relevant to our model backing-up and loading methods. An instance of our test class would have an instance of our model and data classes, as well as objects relevant to performing a test on a model, such as number of epochs and batch size. The test class defines a method to perform testing. An instance of our train class would have an instance

of our model and data classes, as well as objects relevant to performing the training of a model, such as number of epochs, batch size, and the the path to keep Tensorboard logs. The train class defines a public method to perform the training of a model, and defines several Keras elements internally, such as data generators and callbacks.

In Algorithm 5.1, we have the implementation of one of our proposed architectures, which will be thoroughly discussed in Sub-section 5.2.2. This algorithm illustrates how defining complex architectures is straightforward in Keras. In this algorithm, we use several layers pre-defined by Keras, such as 2D convolutional layers (Conv2D), 2D max pooling layers (MaxPooling2D), etc. We also have several pre-defined activation functions and optimizers at our disposal.

5.2.2 Experiments

We propose a total of 11 experiments – from A to K – with experiments A to J being performed on our custom architecture, and experiment K being performed on the Inception-V3 architecture.

For all proposed architectures, we used the Rectified Linear Unit (ReLU) (NAIR; HINTON, 2010) as activation function. We also normalize the input images in the training and test datasets for all proposed architectures, as described in Section 4.2.5. For architectures A to J, we resize the images to 256×256. For the Inception-V3 architecture, since we use a pre-trained model, we are unable to change properties such as input size, and therefore we resize the images to 299×299 to match the expected dimension. Reducing the input image size is also important so that we may fit a reasonable quantity of images, say 32 images, in a batch without running out of Graphics Processing Unit (GPU) memory. We train our CNN on a NVIDIA GeForce GTX 1080 Ti, which gives us 11 GB of GPU memory. Though downscaling implies in losing information, it also reduces the amount of weights to be learnt, easing convergence. Simply resizing the input images is straightforward, and has been used before by ImageNet contestants (SIMONYAN; ZISSERMAN, 2014b) with a similar, 224×224, downsized resolution. Radiographs do not necessarily have the same height and width, and it is very important not to alter the aspect ratio of a radiograph, since this may alter the properties rated by radiologists, such as the relative size of the epiphyses of the phalanges to the size of adjacent metaphyses. Therefore, we downscale our radiographs, maintaining the aspect ratio, in a manner that the largest dimension equals to our target size, 256. The smaller dimension, is then padded

Algorithm 5.1: Implementation of the architecture proposed in experiment B

```
image_input = Input(shape=(image_size[0],
    image_size[1],1))
gender_input = Input(shape=(2,))
num_filters = [16,32,64]
filter_size = (3, 3)
current_layer = image_input

for num_filter in num_filters:
    current_layer = Conv2D(num_filter, filter_size,
       padding='same')(current_layer)
    current_layer = InstanceNormalization(axis=3)
        (current_layer)
    current_layer = Activation('relu')(current_layer)

    last_layer = current_layer

    current_layer = Conv2D(num_filter, filter_size,
       padding='same')(current_layer)
    current_layer = InstanceNormalization(axis=3)
        (current_layer)
    current_layer = Activation('relu')(current_layer)

    current_layer = Conv2D(num_filter, filter_size,
       padding='same')(current_layer)
    current_layer = InstanceNormalization(axis=3)
        (current_layer)
    current_layer = Add()([last_layer,current_layer])
    current_layer = Activation('relu')(current_layer)

    current_layer = MaxPooling2D(
        pool_size=(2, 2))(current_layer)

current_layer = Flatten()(current_layer)
current_layer = Concatenate(axis=1)
    ([current_layer,gender_input])

current_layer = Dense(1024)(current_layer)
current_layer = Activation('relu')(current_layer)
current_layer = Dense(512)(current_layer)
current_layer = Activation('relu')(current_layer)
output_boneage = Dense(1)(current_layer)

model = Model(inputs=[image_input,
    gender_input],outputs=output_boneage)
model.compile(loss = "mse",
    optimizer = "adam", metrics=["mae"])
```

with zeroes. Alternatives to resizing the images may include:

- Dividing the input image in ROIs, and training on each ROI individually. BAA could then be performed by using the ensemble of individual evaluations. However, this is a complex approach that may not outperform its straightforward counterpart.
- Using a Bulk Synchronization Parallel (BSP) model (WU et al., 2017). This approach consists in splitting the image into smaller patches that are fed to the CNN separately. Then, a padding and normalization technique merges the patches into a single image. The paper was not made available until the end of the challenge, but it could be an improvement to the currently used technique.

Tables 5.1 and 5.2 show the experiments performed on our custom architecture. We proposed these experiments in a way that from A to J, they are progressively more complex – meaning that we add more layers, regularization, increase filter sizes, etc. The value of the changed parameter in each experiment – the dropout value, for instance – was in many cases proposed after a series of attempts with different values, from which only the best value was chosen.

Table 5.1: Experiments with custom architecture - Part 1

	Architecture						
	Conv. layers		Filters per block	Filter size	Regularization		Batch/Instance Normalization
	By block	Total			L2	Dropout	
A	3	9	16, 32, and 64	3×3	–	–	–
B	3	9	16, 32, and 64	3×3	–	–	–
C	3	9	16, 32, and 64	3×3	–	–	–
D	3	9	16, 32, and 64	3×3	–	–	batch
E	3	9	16, 32, and 64	3×3	–	–	instance
F	5	20	16, 32, 64, and 128	3×3	–	–	instance
G	7	35	16, 32, 64, 128, and 256	3×3	–	–	instance
H	7	35	16, 32, 64, 128, and 256	5×5	–	–	instance
I	7	35	16, 32, 64, 128, and 256	5×5	–	0.15	instance
J	7	35	16, 32, 64, 128, and 256	5×5	0.0001	0.15	instance

Table 5.2: Experiments with custom architecture - Part 2

	Optimizer	Data					
	Algo-rithm	Data augmentation					Image size
		Rotation range	Width shift	Height shift	Zoom range	Horizontal flip	
A	SGD	–	–	–	–	no	256×256
B	Adam	–	–	–	–	no	256×256
C	Adam	10	0.05	0.05	0.05	yes	256×256
D	Adam	10	0.05	0.05	0.05	yes	256×256
E	Adam	10	0.05	0.05	0.05	yes	256×256
F	Adam	10	0.05	0.05	0.05	yes	256×256
G	Adam	10	0.05	0.05	0.05	yes	256×256
H	Adam	10	0.05	0.05	0.05	yes	256×256
I	Adam	10	0.05	0.05	0.05	yes	256×256
J	Adam	10	0.05	0.05	0.05	yes	256×256

5.2.3 Results

We trained our custom architecture based on the experiments proposed on Tables 5.1 and 5.2. We ran experiments A to H for 50 epochs, being an epoch a measure of how many times we have taken enough steps to see our whole dataset. Experiments I, J, and K were ran for an additional 100 epochs, for we believed that they, due to their complexity, would be able to learn more if trained longer.

Experiment A failed to converge due to its optimizer algorithm, SGD (see Section 4.2.3), which is highly dependent on fine-tuning its parameters. This is not an issue since Adam is proven to have a better, faster convergence. The final MAE of each experiment is displayed at Table 5.3, and the overall progression of the validation MAE for each experiment can be seen at Figures 5.6 and 5.7.

Table 5.3: Results - Bone Age Assessment Module

	Parameter modified	Validation Mean Average Error	
		Epoch 100	Epoch 150
A	–	*	–
B	Optimizer	≈ 20.47	–
C	Data Augmentation (rotation, width and height shift, zoom, and horizontal flip)	≈ 10.47	–
D	Batch Normalization	≈ 16.61	–
E	Instance Normalization	≈ 12	–
F	Depth and Width	≈ 10.79	–
G	Depth and Width	≈ 16.98	–
H	Filter Size	≈ 10.73	–
I	Dropout	≈ 20.23	≈ 20.33
J	L2 Regularization	≈ 11.54	≈ 9.74

Figure 5.6: MAE x epochs plot for experiments B to H. Source: The authors.

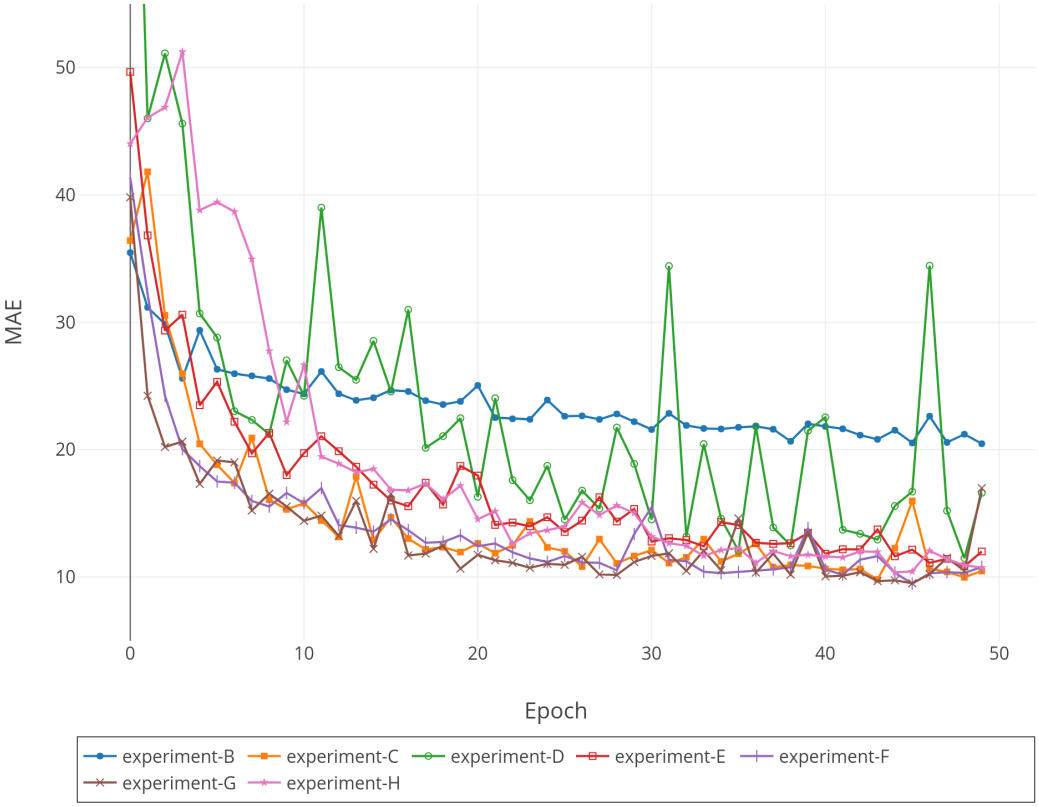
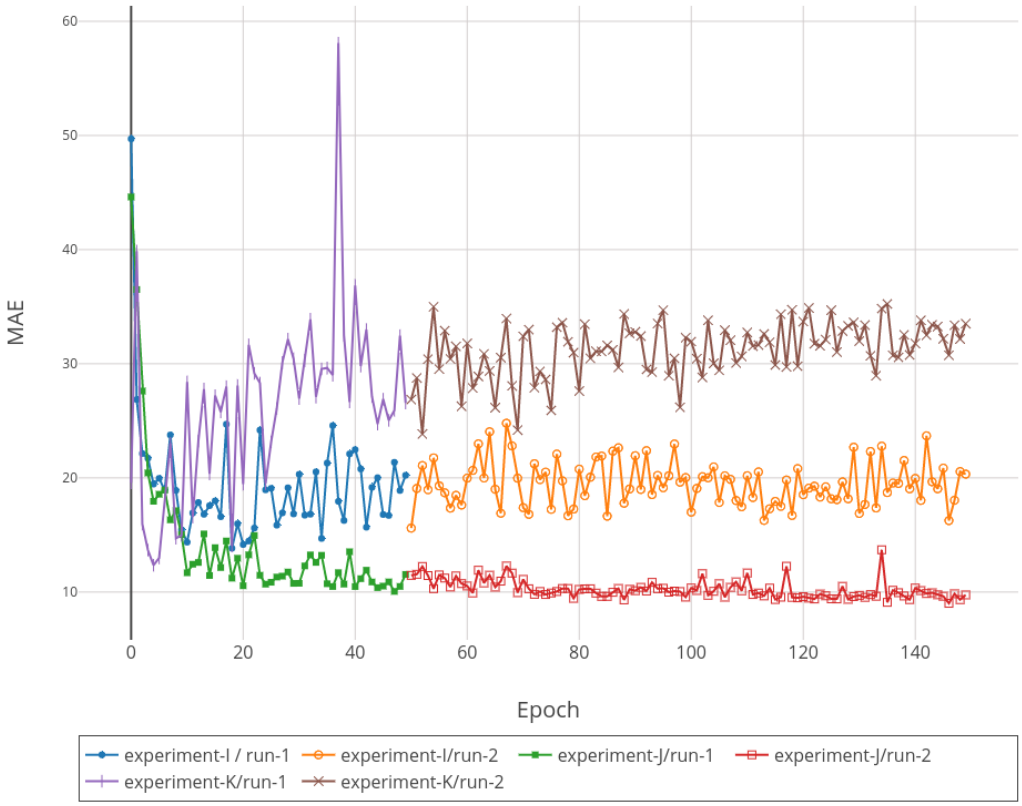


Figure 5.7: MAE x epochs plot for experiments I and J. Source: The authors.



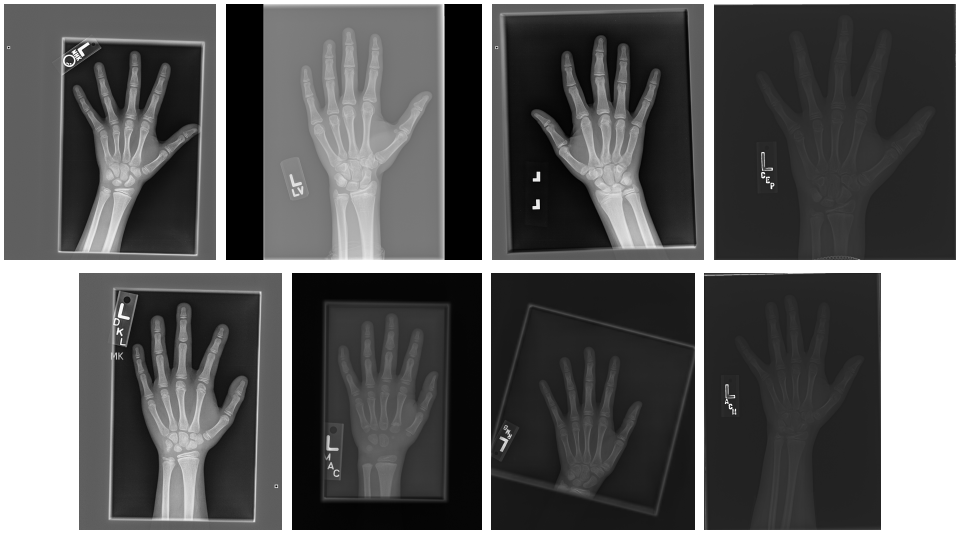
After analyzing our results, we chose experiment J as the best candidate for submission to the RSNA challenge. Since this was a competition, we could only evaluate our results by submitting our inferences to RSNA's website. Therefore, our decisions were not solely based on which network showed the best results, but on which network we believed to be the most robust.

5.2.3.1 RSNA Challenge

We evaluated the accuracy of our network by performing BAA for the 200 radiographs that constitute the test dataset of the RSNA challenge. Some of these images are shown in Figure 5.8. The ground-truth bone age for these radiographs was only known by the organizers of the challenge. The average time for estimating the bone age for each radiograph was approximately $35\ ms$. The complete set of estimates was submitted online to the RSNA contest, and our results achieved a MAE of 6.44 months and a CCC of 0.97. Such CCC value demonstrates *a substantial level of agreement between our predictions and the ground-truth* (MCBRIDE, 2005). With this MAE, we attained the 13^{th}

place among 300 teams worldwide.

Figure 5.8: 8 samples of radiographs available on the test dataset. Source: RSNA.



There were several approaches to solve this problem – some of those made public by the competitors –, varying from fine-tuning famous CNNs to complex, deep CNNs. Most of the best achieving competitors trained with larger resolutions (512×512) than us – something we were unable to do due to hardware limitations. We believe our results could be further improved by using larger resolutions.

The Digital Hand Atlas (CAO; HUANG; PIETKA, 2000), prepared by researchers from the University of Southern California, is the most comprehensive dataset of hand radiographs with bone age estimates produced by two raters. For this dataset, the inter-rater variability can be summarized by an overall Root-Mean-Square Error (RMSE) of 0.59 years(0.57 years for males, and 0.54 years for females), and 0.66 years for children between 5 and 18 years old. A more recent study (KIM; OH; SHIN, 2008) reports an inter-rater variability of 0.51 ± 0.44 years with the use of the GP method. Our results obtained a MAE of 0.536 years (6.44 months), which is in accordance with results produced by expert radiologists.

For comparison, the average error in the estimates obtained with the recent deep-learning-based system described in Lee et al. (2017) is 0.82 years for males and 0.93 years for females. The commercial software BoneXpert[1] has an average error of 0.72 years. Our results surpasses these systems.

The best results for the RSNA challenge were obtained by the system described by Iglovikov et al. (IGLOVIKOV et al., 2017), which obtained a MAE of 4.97 months.

[1] https://www.bonexpert.com/products/the-bonexpert-product

Despite the use of three different CNNs and the need for manual intervention to create training sets for the additional networks, the average results obtained by the solution described by Iglovikov et al. was only 45 days closer to the ground truth than ours. Such difference does not seem to be significant from a clinical perspective. Furthermore, given the inherent imprecision of the GP and TW2 methods, the accuracy of our solution can be considered comparable to Iglovikov et al.'s. Our solution, however, can be easily applied to different datasets (*e.g.*, to handle different ethnic groups) and requires less training time.

5.2.4 Demonstration

For demonstration purposes, we developed a website[2] where a user can upload a radiograph and receive the corresponding bone age assessed by our algorithm. This website's front-end was developed using Angular – an open-source front-end web application framework – and our back-end was implemented with Django – a high-level Python web framework. Our server handles the uploading of images, the serving of static content, and the actual BAA.

In Figure 5.9, we can see a BAA performed by our website. The user must fill the "Patient Info" section, by providing a standard bone age radiograph and the patient's gender. In the bottom-most section, "Result", we can see the original radiograph overlaid by its class-activation map, highlighting the regions that influenced the most the BAA for the given radiograph (see Chapter 6). We can also see the nearest reference radiograph for the predicted bone age – extracted from the Greulich and Pyle atlas (GREULICH; PYLE, 1959).

[2] https://iarahealth.com/boneage/

Figure 5.9: On-line automated BAA using our technique. Source: The authors.

6 DISCUSSION AND VISUALIZATION OF RELEVANT FEATURES

CNNs, particularly end-to-end systems as ours, may excel in performance, but suffer a big penalty to result interpretability when compared to traditional computer vision algorithms, for instance. The more specific, hand-crafted features or guidance a learning system receives, the more straightforward it is to understand why it succeeds or fails for each given input. In order to better understand the behaviour of our CNN and its results, we use a technique known as Gradient-weighted Class Activation Mapping (Selvaraju et al., 2016) (Grad-CAM), implemented by the Keras Visualization Toolkit (KOTIKALAPUDI; CONTRIBUTORS, 2017).

Grad-CAM uses the gradients relative to the upper convolutional layers of our network to produce a map highlighting the regions who most influenced the prediction. The toolkit used underwent a series of changes to accept our input, since our network expects both the gender and the radiograph of the patient, and KerasVIS used images only. We also had to adapt it to work with single-channel images. Our Grad-CAMs use a color map to represent the importance of a region to the BAA, in a manner that the warmer the color a region is, the stronger the influence it had on the inference.

In Figure 6.1, we can see the Grad-CAMs for radiographs of different stages of skeletal maturity for female (upper row) and male (lower row) patients: pre-puberty, early and mid-puberty, late-puberty, and post-puberty – toddlers were left out on purpose, for there are few images for this age range in the training dataset, as seen on Figure 3.2. The radiographs used represent cases where our algorithm predicted the correct bone age or the prediction error was below 6 months, for these cases generate cleaner Grad-CAMs.

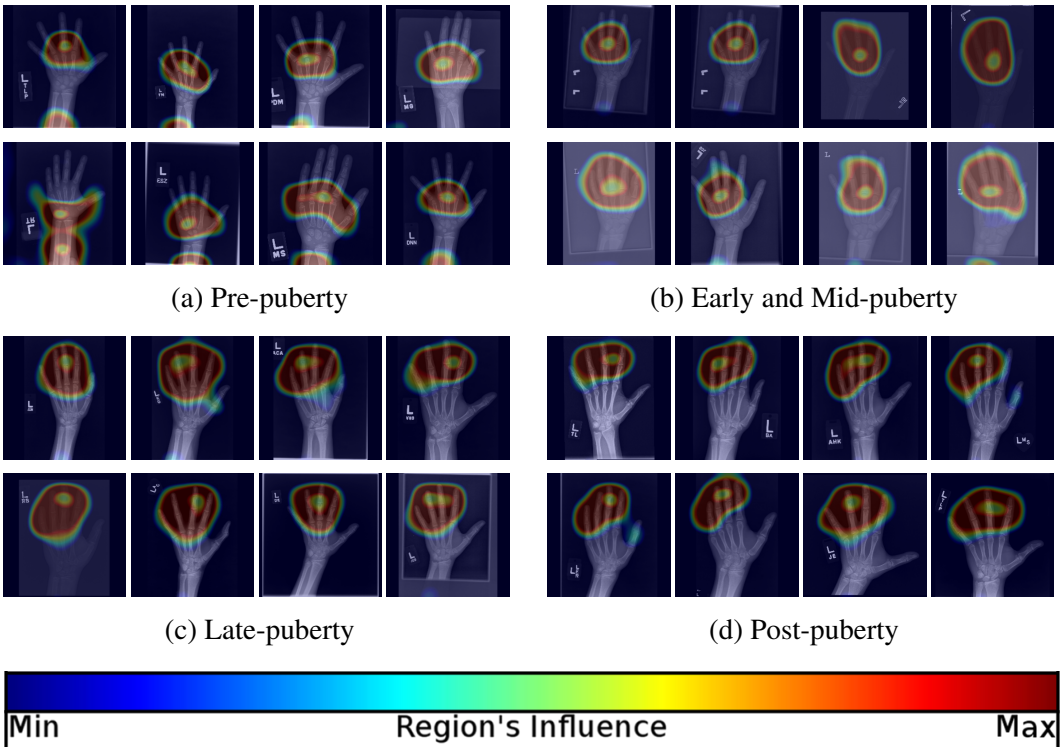
We compared the regions highlighted by our Grad-CAMs to the regions taken into account by radiologists, as detailed in Gilsanz and Ratib (2005), while performing BAA for each stage of skeletal maturity:

- Pre-puberty: In this stage, radiologists primarily compare the size of the *epiphyses* of the phalanges to the size of adjacent *metaphyses*. Our CNN takes a very different approach, deeming the ulna, radius, carpal and metacarpal bones to be the most important structures for BAA in this stage.
- Early and mid-puberty: In this stage, radiologists primarily analyze the size of the epiphyses in the distal and middle phalanges. Our CNN, again, takes a very different approach, being the metacarpal bones and proximal phalanges the most important structures for BAA in this stage.

- Late-puberty: In this stage, radiologists primarily analyze the degree of epiphyseal fusion of the distal phalanges. Our CNN finds the distal phalanges to be important for BAA in this stage, but also finds the middle and proximal phalanges, and metacarpal bones to be important to some extent.
- Post-puberty: In this stage, radiologists primarily analyze the degree of epiphyseal fusion of the ulna and radius bones. Our CNN takes a very different approach, deeming the phalanges alone to be the most important structures for BAA in this stage.

Overall, the regions our CNN analyzes do not match the regions analyzed by radiologists, which could suggest there may be other ways to manually perform BAA.

Figure 6.1: Grad-CAMs generated for each skeletal maturity stage. Source: The authors.



(a) Pre-puberty

(b) Early and Mid-puberty

(c) Late-puberty

(d) Post-puberty

7 CONCLUSION AND FUTURE WORK

We presented an automated approach for bone age assessment based on residual learning. We trained and validated our CNN on a partially-public dataset containing over 12,500 radiographs from the Pediatric Bone Age Challenge (RSNA, 2018), organized by the Radiological Society of North America (RSNA). We evaluated the accuracy of our network by performing BAA for 200 radiographs from the test dataset of the RSNA challenge. Our results achieved a MAE of 6.44 months and a CCC of 0.97, indicating a substantial level of agreement between our predictions and the ground-truth. The accuracy of our solution is similar to the one obtained by expert radiologists and superior to previous automated systems. Although our system obtained for the RSNA challenge dataset a MAE of 45 days bigger than the winner system (IGLOVIKOV et al., 2017), such difference does not seem to be clinically significant. But unlike Iglovikov et al.'s system, ours is an end-to-end solution, can be easily trained with different datasets, and requires less training time.

We have provided a discussion of the most important hand and wrist structures for BAA according to our CNN and compared these with the features observed by expert radiologists. A free on-line bone age assessment service based on our system is available and can be a valuable resource for doctors working in remote areas or in institutions with no BAA experts.

Once our segmentation pipeline is showing good results, we could perform experiments M and K and choose the best performing solution among all experiments. We then propose to evaluate the performance of our algorithm by validating it on a real use case by integrating it to a Picture Archiving and Communication System (PACS), a technology used for storing, retrieving, visualizing and sharing medical images. Such systems are commonly used in hospitals, and would allow us to easily integrate our algorithm to the radiologist's work-flow. We aim to restrain the usage of our algorithm to the work-flow of a small group of radiologists at first, so that we can validate our algorithm and estimate how well it would work on a larger scale. A scientific survey will also be conducted to verify not only the accuracy of our algorithm, but also the impact it has on the radiologist's work-flow, and overall receptiveness of machine learning algorithms to aid in the diagnosis process.

REFERENCES

ABADI, M. et al. **TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems**. 2015. Software available from tensorflow.org. Available from Internet: <https://www.tensorflow.org/>.

ACADEMY, I. D. **Skeletal maturity indicator**. 2016. <https://www.slideshare.net/indiandentalacademy/skeletal-maturity-indicator-2>.

BELL, D. J.; MURPHY, A. e. a. **Computed radiography**. 2017. [Online; accessed 14-November-2017]. Available from Internet: <https://radiopaedia.org/articles/computed-radiography>.

BRADSKI, G. The OpenCV Library. **Dr. Dobb's Journal of Software Tools**, 2000.

BUNCH, P. M. et al. Skeletal development of the hand and wrist: digital bone age companion-a suitable alternative to the greulich and pyle atlas for bone age assessment? **Skeletal radiology**, v. 46, n. 6, p. 785–793, Jun 2017.

CAO, F.; HUANG, H.; PIETKA, E. e. a. Digital hand atlas for web-based bone age assessment: System design and implementation. v. 24, p. 297–307, 04 2000.

CHOLLET, F. et al. **Keras**. [S.l.]: GitHub, 2015. <https://github.com/fchollet/keras>.

COLLOBERT, R.; KAVUKCUOGLU, K.; FARABET, C. Torch7: A matlab-like environment for machine learning. In: **BigLearn, NIPS Workshop**. [S.l.: s.n.], 2011.

DRAKE, F. L. J. **Python 2.7.14 documentation**. [S.l.], 2017. Available from Internet: <https://docs.python.org/2/index.html>.

DUCHI, J.; HAZAN, E.; SINGER, Y. Adaptive subgradient methods for online learning and stochastic optimization. **J. Mach. Learn. Res.**, JMLR.org, v. 12, p. 2121–2159, jul. 2011. ISSN 1532-4435. Available from Internet: <http://dl.acm.org/citation.cfm?id=1953048.2021068>.

ESTEVA, A.; KUPREL, B.; NOVOA, R. A. e. a. Dermatologist-level classification of skin cancer with deep neural networks. **Nature**, Macmillan Publishers Limited, part of Springer Nature. All rights reserved., v. 542, p. 115–, jan. 2017.

FACEBOOK. **Torch implementation of DeepMask and SharpMask**. 2016. Available from Internet: <https://github.com/facebookresearch/deepmask>.

GASKIN, C. et al. **Skeletal Development of the Hand and Wrist: A Radiographic Atlas and Digital Bone Age Companion**. Oxford University Press, USA, 2011. ISBN 9780199782055. Available from Internet: <https://books.google.com.br/books?id=qTe8Y8nFUW8C>.

GILSANZ, V.; RATIB, O. **Hand Bone Age: A Digital Atlas of Skeletal Maturity**. [S.l.]: Springer Berlin Heidelberg, 2005. ISBN 9783540270706.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. Deep learning. In: . [S.l.]: MIT Press, 2016. chp. 6. <http://www.deeplearningbook.org>.

GOODFELLOW, I. et al. Generative adversarial nets. In: GHAHRAMANI, Z. et al. (Ed.). **Advances in Neural Information Processing Systems 27**. Curran Associates, Inc., 2014. p. 2672–2680. Available from Internet: <http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf>.

GREULICH, W. W.; PYLE, S. I. **Radiographic Atlas of Skeletal Development of the Hand and Wrist**. Stanford, CA, USA: Stanford University Press, 1959.

HE, K. et al. Deep Residual Learning for Image Recognition. **ArXiv e-prints**, dec. 2015.

IGLOVIKOV, V. et al. Pediatric bone age assessment using deep convolutional neural networks. **CoRR**, abs/1712.05053, 2017.

INITIATIVE, O. S. **The 3-Clause BSD License**. 2012. Available from Internet: <https://opensource.org/licenses/BSD-3-Clause>.

IOFFE, S.; SZEGEDY, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. **CoRR**, abs/1502.03167, 2015. Available from Internet: <http://arxiv.org/abs/1502.03167>.

KIM, S. Y.; OH, Y. J.; SHIN, J. e. a. Y. Comparison of the greulich-pyle and tanner whitehouse (tw3) methods in bone age assessment. v. 13, p. 50–55, 06 2008.

KINGMA, D. P.; BA, J. Adam: A method for stochastic optimization. **CoRR**, abs/1412.6980, 2014. Available from Internet: <http://arxiv.org/abs/1412.6980>.

KOTIKALAPUDI, R.; CONTRIBUTORS. **keras-vis**. [S.l.]: GitHub, 2017. <https://github.com/raghakot/keras-vis>.

KöRNER, M. et al. Advances in digital radiography: Physical principles and system overview. **RadioGraphics**, v. 27, n. 3, p. 675–686, 2007. PMID: 17495286. Available from Internet: <https://doi.org/10.1148/rg.273065075>.

LEE, H. et al. Fully-automated deep learning system for bone age assessment. **Journal of Digital Imaging**, p. 1–15, 2017.

LI, R.; ZHANG, W.; SUK, H. I. e. a. Deep learning based imaging data completion for improved brain disease diagnosis. **Medical image computing and computer-assisted intervention**, v. 17, n. Pt 3, p. 305–312, 2014.

LIN, T. et al. Microsoft COCO: Common Objects in Context. **ArXiv e-prints**, may 2014.

LITJENS, G. J. S. et al. A survey on deep learning in medical image analysis. **CoRR**, abs/1702.05747, 2017.

MCBRIDE, G. A proposal for strength-of-agreement criteria for lin's concordance correlation coefficient. p. 2005–2062, 01 2005.

MINSKY, M. Steps toward artificial intelligence. **Proceedings of the IRE**, v. 49, p. 8–30, Jan 1961.

MITCHELL, T. M. **Machine Learning**. 1. ed. New York, NY, USA: McGraw-Hill, Inc., 1997. ISBN 0070428077, 9780070428072.

MUGHAL, A. M.; HASSAN, N.; AHMED, A. Bone age assessment methods: A critical review. **Pakistan Journal of Medical Sciences**, v. 30, n. 1, p. 211–215, Jan 2014.

NAIR, V.; HINTON, G. E. Rectified linear units improve restricted boltzmann machines. In: **Proceedings of the 27th International Conference on International Conference on Machine Learning**. USA: Omnipress, 2010. (ICML'10), p. 807–814. ISBN 978-1-60558-907-7. Available from Internet: <http://dl.acm.org/citation.cfm?id=3104322.3104425>.

OESTREICH, A. E. Hand bone age: A digital atlas of skeletal maturity. **RadioGraphics**, v. 25, n. 4, p. 1074–1074, 2005. Available from Internet: <https://doi.org/10.1148/radiographics.25.4.0251074>.

OTSU, N. A threshold selection method from gray-level histograms. **IEEE Transactions on Systems, Man, and Cybernetics**, v. 9, n. 1, p. 62–66, Jan 1979. ISSN 0018-9472.

PINHEIRO, P.; COLLOBERT, R.; DOLLAR, P. Learning to Segment Object Candidates. **ArXiv e-prints**, jun. 2015.

PINHEIRO, P. et al. Learning to Refine Object Segments. **ArXiv e-prints**, mar. 2016.

RASCHKA, S. **About Feature Scaling and Normalization – and the effect of standardization for machine learning algorithms**. 2014. <http://sebastianraschka.com/Articles/2014_about_feature_scaling.html>.

RONNEBERGER, O.; FISCHER, P.; BROX, T. U-net: Convolutional networks for biomedical image segmentation. **CoRR**, abs/1505.04597, 2015.

RSNA. **RSNA Pediatric Boneage Challenge**. 2018. [Online; accessed 30-May-2018]. Available from Internet: <http://rsnachallenges.cloudapp.net/competitions/4>.

RUDER, S. An overview of gradient descent optimization algorithms. **CoRR**, abs/1609.04747, 2016. Available from Internet: <http://arxiv.org/abs/1609.04747>.

RUSSAKOVSKY, O. et al. ImageNet Large Scale Visual Recognition Challenge. **International Journal of Computer Vision (IJCV)**, v. 115, n. 3, p. 211–252, 2015.

SALAM, H. e. a. **Delayed bone age**. 2014. [Online; accessed 26-October-2017]. Available from Internet: <https://radiopaedia.org/articles/delayed-bone-age-1>.

Mean absolute error. In: SAMMUT, C.; WEBB, G. I. (Ed.). **Encyclopedia of Machine Learning**. Boston, MA: Springer US, 2010. p. 652–652. ISBN 978-0-387-30164-8. Available from Internet: <https://doi.org/10.1007/978-0-387-30164-8_525>.

SATOH, M. Bone age: assessment methods and clinical applications. **Clinical Pediatric Endocrinology**, v. 24, n. 3, p. 143–152, Oct 2015.

SCHINDELIN, J. et al. Fiji: an open-source platform for biological-image analysis. **Nature Methods**, Nature Publishing Group, v. 9, n. 7, p. 676–682, jul. 2012. ISSN 15487091.

SCHMIDHUBER, J. Deep Learning in Neural Networks: An Overview. **ArXiv e-prints**, abr. 2014.

SCHNEIDER, C. A.; RASBAND, W. S.; ELICEIRI, K. W. Nih image to imagej: 25 years of image analysis. **Nature Methods**, Nature Publishing Group, v. 9, n. 7, p. 671–675, jul. 2012. ISSN 15487091.

Selvaraju, R. R. et al. Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization. **ArXiv e-prints**, oct. 2016.

SEOK, J. et al. Automated classification system for bone age x-ray images. In: **2012 IEEE International Conference on Systems, Man, and Cybernetics (SMC)**. [S.l.: s.n.], 2012. p. 208–213. ISSN 1062-922X.

SHETTY, A.; SALAM, H. e. a. **Increased bone age**. 2014. [Online; accessed 26-October-2017]. Available from Internet: <https://radiopaedia.org/articles/increased-bone-age>.

SIMONYAN, K.; ZISSERMAN, A. Very deep convolutional networks for large-scale image recognition. **CoRR**, abs/1409.1556, 2014. Available from Internet: <http://arxiv.org/abs/1409.1556>.

SIMONYAN, K.; ZISSERMAN, A. Very deep convolutional networks for large-scale image recognition. **CoRR**, abs/1409.1556, 2014. Available from Internet: <http://arxiv.org/abs/1409.1556>.

SOMKANTHA, K.; THEERA-UMPON, N.; AUEPHANWIRIYAKUL, S. Bone age assessment in young children using automatic carpal bone feature extraction and support vector regression. **Journal of Digital Imaging**, v. 24, p. 1044–1058, 2011.

SZEGEDY, C. et al. Going deeper with convolutions. In: **2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2015. p. 1–9. ISSN 1063-6919.

SZEGEDY, C. et al. Rethinking the Inception Architecture for Computer Vision. **ArXiv e-prints**, dec. 2015.

TANNER, J. et al. Book. **Assessment of skeletal maturity and prediction of adult height (TW2 method)**. [S.l.]: Academic Press London ; New York, 1975. vii, 99 p. : p. ISBN 0126833508.

TANNER, J. M.; WHITEHOUSE, R. J. **A New System for Estimating Skeletal Maturity from the Hand and Wrist: With Standards Derived from a Study of 2,600 Healthy British Children**. [S.l.]: International Children's Centre, 1959.

THODBERG, H. H. et al. The bonexpert method for automated determination of skeletal maturity. **IEEE Transactions on Medical Imaging**, v. 28, n. 1, p. 52–66, Jan 2009. ISSN 0278-0062.

TIELEMAN, T.; HINTON, G. **Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude**. 2012. COURSERA: Neural Networks for Machine Learning.

ULYANOV, D.; VEDALDI, A.; LEMPITSKY, V. S. Instance normalization: The missing ingredient for fast stylization. **CoRR**, abs/1607.08022, 2016. Available from Internet: <http://arxiv.org/abs/1607.08022>.

WU, S. et al. A new approach to compute cnns for extremely large images. In: **Proceedings of the 2017 ACM on Conference on Information and Knowledge Management**. [S.l.]: ACM, 2017. (CIKM '17), p. 39–48. ISBN 978-1-4503-4918-5.

YOSINSKI, J. et al. How transferable are features in deep neural networks? **CoRR**, abs/1411.1792, 2014. Available from Internet: <http://arxiv.org/abs/1411.1792>.

ZHANG, A.; GERTYCH, A.; LIU, B. J. Automatic bone age assessment for young children from newborn to 7-year-old using carpal bones. v. 31, p. 299–310, 06 2007.

ZHANG, J.; LIN, F.; DING, X. Maturation disparity between hand-wrist bones in a chinese sample of normal children: An analysis based on automatic bonexpert and manual greulich and pyle atlas assessment. **Korean journal of radiology**, v. 17, n. 3, p. 435—442, 2016. ISSN 1229-6929.

Deep learning for medical image analysis. In: ZHOU, S. K.; GREENSPAN, H.; SHEN, D. (Ed.). Academic Press, 2017. p. iv –. ISBN 978-0-12-810408-8. Available from Internet: <https://www.sciencedirect.com/science/article/pii/B9780128104088000274>.

ZHOU, Y.; CHELLAPPA, R. Computation of optical flow using a neural network. In: **IEEE 1988 International Conference on Neural Networks**. [S.l.: s.n.], 1988. p. 71–78 vol.2.

ZHU, J. et al. Unpaired image-to-image translation using cycle-consistent adversarial networks. **CoRR**, abs/1703.10593, 2017. Available from Internet: <http://arxiv.org/abs/1703.10593>.