Computing Methods for Experimental Physics and Data Analysis
RSNA Bone Age Assessment Challenge

Students: Giuseppe Fanciulli, Giuseppe Antonio Motisi

Academic year 2023/24

Abstract

This project aims to determine through machine learning methods the bone age from hand radiographs of patients aged 0 to 228 months. Informations about the challenge can be found on the RSNA website [5]. After some preliminary operations on the raw RSNA dataset (merge and split to remove overlaps), we continued the work of dataset preparation; in particular, we decided to preprocess all images and increase the number of images with appropriate image augmentation techniques. Then, we used our custom neural network model, called `rVGG16-L2`, to extract a single output. This network was used with both no augmented and augmented image datasets and results were compared. After routine fine-tuning operation, we achieved a precision of about 9 months.

1 Introduction

Bone age assessment is the standard method used from doctors in order to estimate the maturity of a child's skeletal system. It simply consists in taking an X-ray image of the wrist, hand and fingers of the subject. The wrist was chosen because its growth can represent the whole body bone development and the radiation damage to the human body is the least when taking X-rays. The traditional bone age recognition methods makes use of a bone age standard atlas, or alternatively of a scoring method. The atlas method consists in comparing the acquired X-ray image with the standard bone age atlas to infer the bone age. The scoring method requires the doctor to divide the development status of each bone in the hand into different grades, and then evaluate the corresponding grades and scores of different bones. The final score of each X-ray image is the sum of all scores and it can be used to infer the bone age via the median curve of bone maturity score. Of course both these methods are subjected to human error since the evaluation depends completely on the doctors' skills.

Bone age can reflect the level and maturity of human growth and development. Bone age assessment is widely used in clinical medicine, forensic medicine, sports medicine and other fields. In clinical medicine, skeletal development can lead to the diagnosis of endocrine, developmental and nutritional disorders. Through bone age, it is possible to determine the appropriate time for orthopaedic surgery (like teeth or nasal cavity) and provide a basis for predicting the adult height of the patient. In forensic science, bone age can estimate the real birth date of an individual and provide legal basis for criminal identification. Moreover, the information about bone maturity can guide the selection of athletes more scientifically.

As mentioned above, bone age assessment is a technique prone to human error, therefore, with the popularization and development of computer technology, machine learning-based bone age prediction has become a research hotspot in recent years. First of all, machine learning solves the problems of subjective factors linked to the doctors' interpretation, and at the same time reduces the prediction time. From 2007, a lot of ML algorithms for bone age assessment were developed, improving precision over and over and reaching an impressive result of a mean absolute error of 5.46 months. [7]

In the following sections we are going to present the RSNA dataset and the manipulations that we applied to it in order to make it readable and suitable for a Convolutional Neural Network to predict bone age of pediatric hands. We also present our CNN architecture and the results it gave back.

2 Methods

2.1 Dataset

2.1.1 Overview

The available RSNA dataset contains a total of 14036 left hand X-ray images; originally divided in 12611 training images (89.8%), 800 validation images (5.7%)

and 625 test images (4.5%). The X-ray images can be either digital or analogic, so their features, such as size, contrast, brightness, field of view, etc., are not homogeneous and differ from image to image. Moreover, attached to both training and validation/test sets, there is a csv file with gender and true bone age information for each image. Therefore we knew that gender speaking the images were distributed with a little higher percentage of males (approximately 53% against 47% of females), as shown in Figure 1.

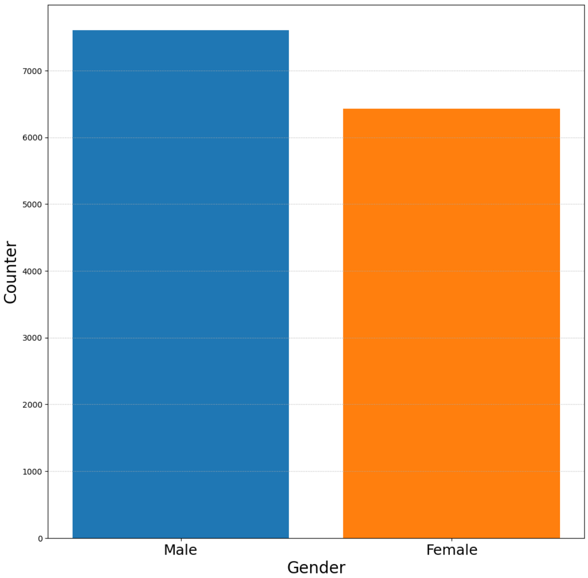

Figure 1: Bar plot of gender distributions in the dataset.

We decided to expand the validation and test dataset in order to have a more solid validation step and broader statistics on the test predictions. What we did was divide the dataset as follows: 9824 training images (70%), 2816 validation images (20%) and 1396 test images (10%), as shown in Figure 2.

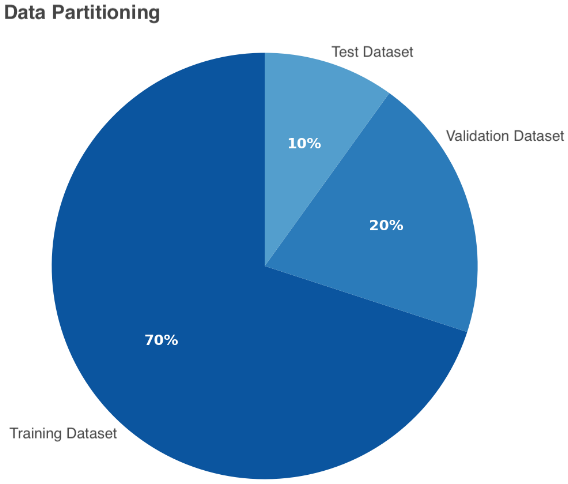

Figure 2: Pie chart for dataset partitioning.

Then, since the images age distribution was not uniform, as shown in Figure 3, we decided to augment them with a ADASYN (Adaptive Synthetic Sampling) algorithm. After this step we would have a balanced training dataset with 27170 images uniformly age distributed from 0 to 228 months. In Figure 4 it is shown the original age distribution for the dataset (before augmentation) and the one after augmentation.

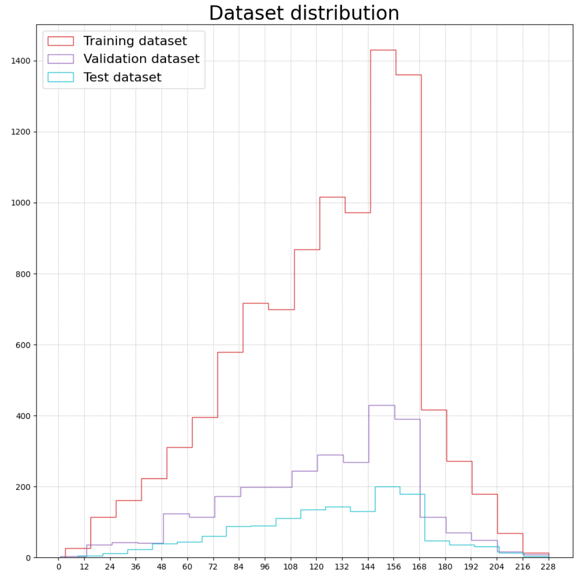

Figure 3: Age distribution histogram in months for training, validation and test datasets.

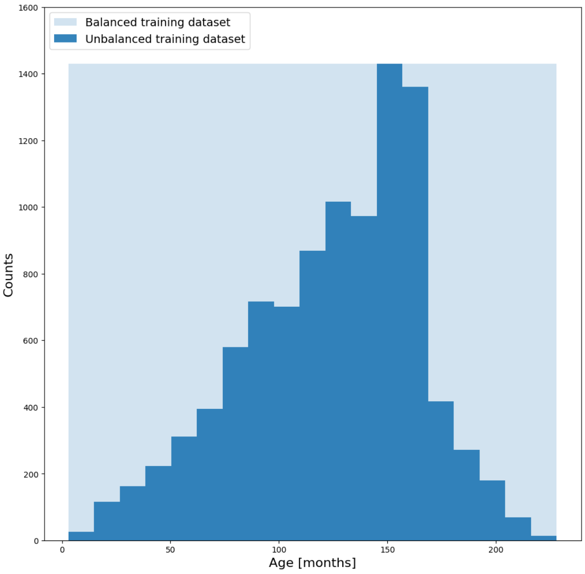

Figure 4: Age distribution histogram in months for original training dataset (darker blue) and for augmented training dataset (lighter blue). Each bin is 12 months wide.

For more details about methods used for this preliminary work, see the documentation in the section RSNA (https://github.com/giuseppeantoniomotisi/boneageassessment/blob/main/documentation/markdown/RSNA.md).

However, we do not recommend working on the raw RSNA dataset and for this reason, we make our dataset available through the following link: https://drive.google.com/drive/folders/1zNPHIJymBkvtQQkJgFwa-7K-x1XH2qdp?usp=share_link.

2.1.2 Augmentation and Preprocessing

As stated above, for the augmentation step we balanced the training dataset. To do this we took the training images and we performed a rotation of a random angle between -20 and 20 degrees around the vertical axis. Each age bin is augmented until is reached the amount of images corresponding to the maximum number of occurrences in the age histogram, so that the new histogram is basically a rectangle.

Once the augmentation step was done we wanted to perform a preprocessing step, that has two main purposes: one is to reduce the dataset size and the second is to clean as much as possible the images (a pseudo-segmentation), making the background black and our subjects (the hands) shiny, i.e. with a higher level of gray. This expedient would create a dataset that would be more readable for our deep learning networks. At first we tried the hand segmentation with a K-means algorithm: we selected three different image regions (background, bone and soft tissue) that should have been discriminated the hand from the background but this method didn't really worked, since it mistook some shady bone regions for background regions. We then switched to the Google mediapipe package [3] to try and detect hands in our images. Whenever the IA algorithm detects one hand, it saves 21 pixels in the image called landmarks. Each of these pixels corresponds to a specific hand zone, so by choosing the top left and the bottom right landmark we can make a bounding box for the hand and crop the image deleting what's outside the box. Then whether the hand is detected or not, the image is further processed with the following operations:

- **Image windowing**: an algorithm finds the leftmost peak (i.e. the darker one) in the image histogram and it puts to zero every pixel with an intensity lower than that corresponding to the bin at the right base of the peak.

- **Histogram equalization**: the image histogram gets equalized.

- **Squaring**: the image is squared, the greater dimension of the image is the chosen side of the square.

- **Resize**: finally the image gets resized to a 399x399 image.

After all these operations, most images get brighter in the hand region and darker in the outer region. The images are ready to be read from some deep learning network to predict the hand bone age.

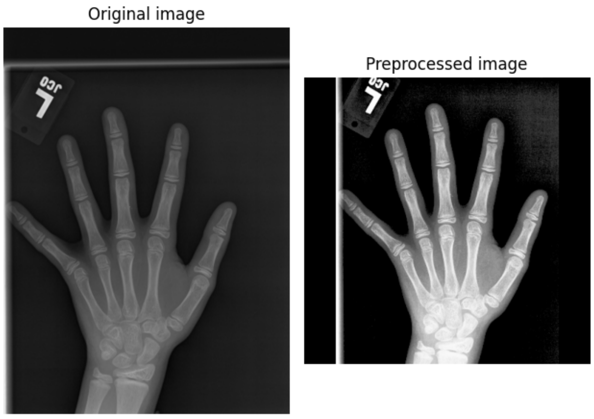

Figure 5: Original image and preprocessed version (with successful cropping step).

2.1.3 Outliers

Looking at the raw RSNA dataset, the first problem is the images pronounced heterogeneity that does not allow Mediapipe to find every hand and so almost a fifth of the images is uncropped, as shown in Figure 6. It is also possible to find images that certainly do not represent normality in the field of bone age assessment. In fact, there are images present that have obvious malformations in terms of anatomy. Some glaring examples of outliers are shown below in Figure 8 and Figure 7.

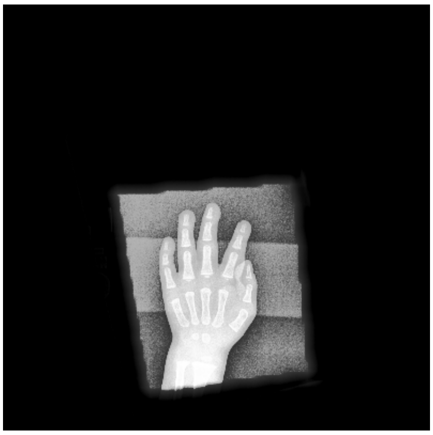

Figure 6: Failed crop of the preprocessing alghorithm. 20% of all the images are uncropped because Mediapipe couldn't find any hand in the image.

Working with another dataset, where there are less outliers, maybe better results could be achieved. For

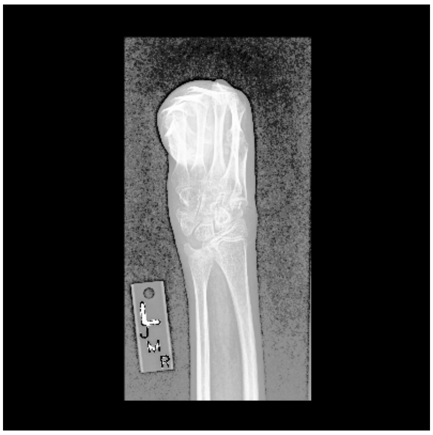

Figure 7: In this case the image is not cropped because the hand is closed.

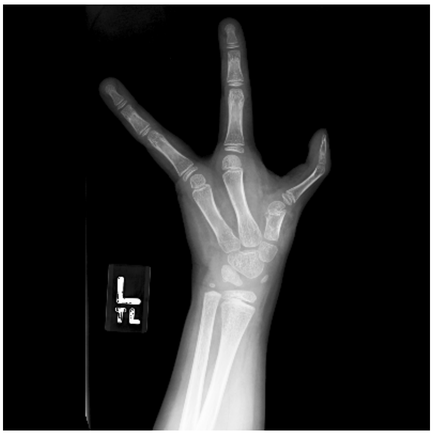

Figure 8: Since the hand is malformed it could not be cropped correctly.

example, collecting all digital images could significantly reduce the number of outliers in the dataset. Or in a more technical way, find an RX protocol that is able to obtain images that span the same dynamic range.

2.2 Model

Before moving on to the details of the network, some premises need to be made. First of all, it is necessary to specify that the problem has been treated as a regression problem in which, starting from a matrix-form input, it was possible to extract a single feature representing the age. Thanks to this assumption, we did not considered necessary to perform any form of label normalization (min-max, z-score, etc.). We also did not consider gender effects, although it has been shown in the literature [6] how influential they can be, especially in pubertal and pre-pubertal ages. In fact, estimating bone age in pediatric age is certainly not an easily solvable problem since the variables contributing to its determination are multiple.

The underlying idea of the model architecture is very simple: we essentially wanted to combine the feature extraction capabilities of a CNN with the predictive capacity of a linear regression model. Furthermore, in accordance with what has been said, given the complexity of the problem, constructing a simple convolutional layers for feature extraction cannot be considered a good approach to the problem. For this reason, we decided to use a VGG16 for the convolutional part and append a regression head to it. We called this simple model `rVGG16`. In practice, `rVGG16` is the grand-father of our final model; indeed, after some attempts we decided to discard this model for two reasons:

- It was really slow and it took a lot of time for training[1].

- It was not so good in generalization.

So we improved the model adding some dropout layers. However, this small modification did not bring significant changes to our results. So we decided to use L2 regularization because it is useful when training a neural network on a limited or noisy dataset in which overfitting occurs. By integrating L2 regularization into the model training, we can control the complexity of the network and improve its ability to generalize unseen data during training. Thanks to this change we obtained good results, shown in **Results**, and the architecture in presented in Figure 9.

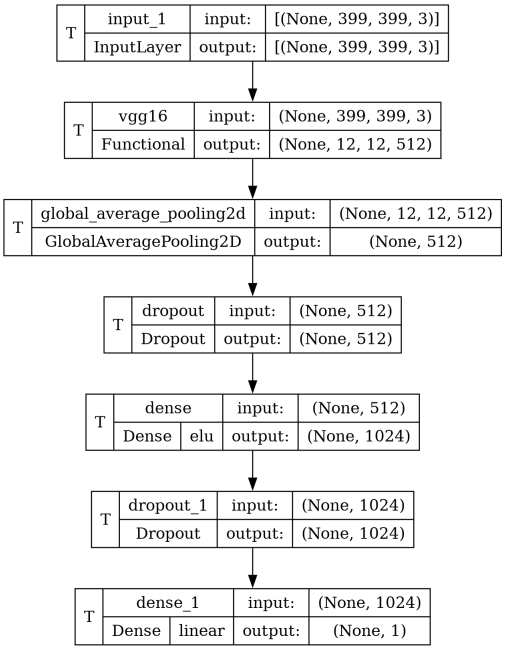
T | input_1 | input: | [(None, 399, 399, 3)]
InputLayer | output: | [(None, 399, 399, 3)]
T | vgg16 | input: | (None, 399, 399, 3)
Functional | output: | (None, 12, 12, 512)
T | global_average_pooling2d | input: | (None, 12, 12, 512)
GlobalAveragePooling2D | output: | (None, 512)
T | dropout | input: | (None, 512)
Dropout | output: | (None, 512)
T | dense | input: | (None, 512)
Dense | elu | output: | (None, 1024)
T | dropout_1 | input: | (None, 1024)
Dropout | output: | (None, 1024)
T | dense_1 | input: | (None, 1024)
Dense | linear | output: | (None, 1)

Figure 9: Graph of a model. A full version is shown in Appendix A.

Some specifications need to be made. In the first place,

[1] We trained our models on Google Colab and later on Kaggle, which imposed a limit on the number of GPU usage hours per week that we could not exceed.

it is obvious that with another dataset maybe this model may not work anymore. However, starting from this architecture, it is straightforward to implement some transfer learning applications by leveraging the `ImageNet` weights. Secondly, it is important to emphasize that the choice of a network like VGG16 is also driven by the limited computational resources at our disposal. In fact, to implement and train the winning models of the challenge would require access to a GPU for more than two days[2]. In the end, the most evident problem: the explainability; with this model, we do not know what features are extracted by CNN. To solve this problem, some attention layers could be added to the network and try to use the attention map to find out what can be improved[3].

3 Results

3.1 Training

Now we could proceed with training of our model. We decided to create a class named `BoneAgeAssessment()`, where are all functions are nested to provide a very-high level experience[4]. Furthermore, this approach has allowed us to shorten the hyperparameter optimization process.

```python
from model import BoneAgeAssessment
from model import BaaModel as Model

LR = 1e-05 # Learning rate
L2 = 1e-04 # Regularization factor
BATCH_SIZE = (32, 32, 1396) # Batch size
EPOCHS = 20 # Number of epochs

# Updates hyperparameters
baa = BoneAgeAssessment()
baa.__update_batch_size__(BATCH_SIZE)
baa.__update_epochs__(EPOCHS)
baa.__update_lr__(LR)

# Show info
baa.__show_info__()

# Create the model
model = Model.vgg16regression_l2(L2)

# Compile the model
baa.compiler(model)

# Training the model
baa.training_evaluation(model)

# Test the model with best weights
WEIGHTS = 'best_model.keras'
PATH_TO_WEIGHTS = os.path.join("..", WEIGHTS)
baa.model_evaluation(PATH_TO_WEIGHTS)
```

3.1.1 Metrics

In order to evaluate the training and validation steps we used two different metrics: the first one is the common Mean Absolute Error (MAE) and the second one is the Pearson correlation coefficient R^2, which measures the linear relationship between two variables, ranging from -1 to +1. The R^2 used for training evaluation is defined as follows:

$$\mathrm{R}^2 = 1 - \frac{\sum_{i=1}^{n}(\tilde{x}_i - x_i)^2}{\sum_{i=1}^{n}(x_i - \bar{x})^2} \qquad (1)$$

where x_i is predicted age in months, $\tilde{x}_i$ is the real age in months and $\bar{x}$ is the mean of measured values.

To evaluate predictions on test dataset we also used two metrics: Mean Absolute Error (MAE) and Mean Absolute Deviance (MAD). This last metric was also used in the RSNA challenge to evaluate the presented network performances and we wanted to have comparable results. The MAD for a set of measurements $X = \{x_1, x_2, ..., x_n\}$ is defined as follows:

$$\mathrm{MAD} = \frac{1}{n}\sum_{i=1}^{n}|x_i - \overline{x}| \qquad (2)$$

Where $\overline{x}$ is the mean of the X set and x_i in our case are the prediction errors $y_{true} - y_{pred}$.

3.1.2 Case a

After conducting preliminary training, we realized that utilizing Adam's default learning rate wasn't optimal, as it failed to generalize well to the validation dataset. Consequently, we opted to initiate training with a learning rate of 0.0001 for this reason. Additionally, other parameters were initially selected based on common values found in literature.

```
LR = 1e-04
L2 = 1e-03
BATCH_SIZE = (32,32,1396)
EPOCHS = 20
BALANCED = True
```

Despite observing symptoms of overfitting by the eighth epoch, the outcomes of this initial case are showcased in the Table 1.

MAE (months)	9.0
MAD (months)	9.0

Table 1: Mean absolute error (MAE) and mean absolute deviation (MAD) in months for case a.

We can also see that errors distribution is quite symmetrical because error range is (-39.9, 39.1) months.

[2] Considering the winning model, the regression InceptionV3 model was trained for about 50 hours on GPUs!

[3] The model `rVGG16-L2-ATN` was created but we did not evaluate its performance for time limit.

[4] Our future purpose is that also a doctor could use this simple API.

3.1.3 Case b

To prevent overfitting, we decided to reduce learning rate to 0.00001.

```
LR = 1e-05
L2 = 1e-03
BATCH_SIZE = (32,32,1396)
EPOCHS = 20
BALANCED = True
```

With these updated parameters, we achieve improved results in mitigating overfitting. Moreover, the distribution of errors remains symmetrical. Results in terms of MAE and MAE in months are reported in Table 2.

MAE (months)	9.3
MAD (months)	9.2

Table 2: Mean absolute error (MAE) and mean absolute deviation (MAD) in months for case b.

3.1.4 Case c

Through case c, we wanted to evaluate how batch size could affect training. This attempt was made because it is recommended to decrease the batch size to enhance the model's generalization capabilities.

```
LR = 1e-05
L2 = 1e-03
BATCH_SIZE = (16,32,1396)
EPOCHS = 20
BALANCED = True
```

However, following a series of attempts, we were unable to train the network due to an abnormal blockage during the run[5].

3.1.5 Case d

For this case, we wanted to evaluate the model performance without a balanced and augmented dataset, so as input we used the original train dataset (9824 images). The hyperparameters were kept as those in case *a*:

```
LR = 1e-04
L2 = 1e-03
BATCH_SIZE = (32,32,1396)
EPOCHS = 20
BALANCED = False
```

The loss function and the validation loss one decreased together without much overfitting, only some oscillating value for the validation. In Table 3 are the results for the prediction using the test dataset.

MAE (months)	10.3
MAD (months)	10.0

Table 3: Mean absolute error (MAE) and mean absolute deviation (MAD) in months for case *d*.

Without augmentation we lost a month of accuracy but the model is still usable. The error distribution is fairly symmetric with respect to zero and the error range is (-65.4, 38.0) months.

3.1.6 Case e

In the last case we decided to set a decreasing learning rate with the epoch number, following the power rule:

$$y(x) = y(1) \cdot 0.95^x \tag{3}$$

Where y is the learning rate value and x is the number of epochs. With this method we tried to let the network learn more slowly with each epoch passing, to prevent overfitting. Listed below are the hyperparameters chosen for this specific training run:

```
LR = 1e-03 to 7.1075e-10
L2 = 1e-04
BATCH_SIZE = (32,32,1396)
EPOCHS = 30
BALANCED = True
```

Unfortunately the method didn't really work, since it started overfitting at epoch five and then the gap between loss and validation loss increased with each epoch. In Table 4 we show the prediction results obtained with this configuration. With an error distribution that is not symmetrical to the zero value (has more value of positive errors, i.e. more overestimated predictions); however the error range is (-65.5, 52.7) months.

MAE (months)	11.5
MAD (months)	11.4

Table 4: Mean absolute error (MAE) and mean absolute deviation (MAD) in months for case *e*.

3.2 Model evaluation

By analyzing both training and test results, it becomes evident that the most favorable outcomes are obtained in case b. Such conclusion stems from the analysis of the obtained results. Specifically, to determine this, we initially observed the trends of the loss vs. epoch graphs to prevent overfitting, Figure 10; subsequently, we evaluated the model's behavior on the test dataset, favoring results that exhibit: 1) a symmetric distribution of the error, 2) the smallest value of MAD (Mean

[5] Approximately between the ninth and fifteenth epoch.

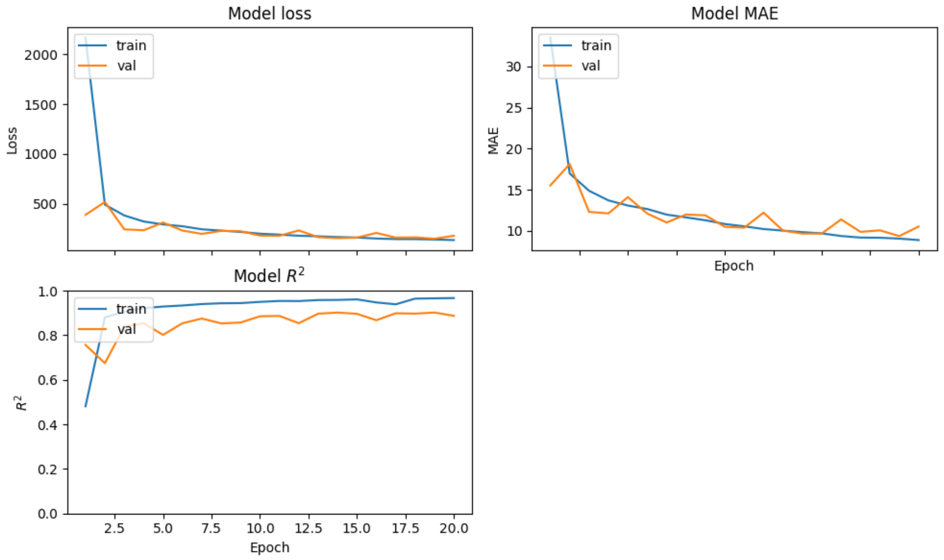

Figure 10: Case b: training evaluation is illustrated through three plots. In the top left, a plot displays the loss versus epoch. In the top right, another plot shows the mean absolute error (MAE) versus epoch. In the bottom left, a plot demonstrates the Pearson correlation coefficient versus epoch. In all plots, training data is represented in blue, and validation data in orange.

Absolute Deviation), and 3) linear adaptability to the extracted data, Figure 12.

As depicted in Figure 10, the loss vs. epoch plot illustrates that the model generalizes effectively on the validation dataset. A similar trend is evident in the MAE vs. epoch plot. Additionally, the R^2 vs. epoch plot reaffirms the observations made in the loss and MAE graphs. Finally, we examine the predictions made on the test dataset, depicted in Figure 12. We chose to plot real age versus predicted age and observed a linear trend. However, towards the graph origin, the line fails to accurately describe predictions, possibly due to a lack of information in that age range. To provide more information about this behavior, we decided to fit our data with a line according to the following equation:

$$y(x) = mx + q$$

where q here represents a bias in data. What we found is shown in Table 5.

	R^2	m	q (months)
a	0.92	0.908 ± 0.007	11.4 ± 0.9
b	0.92	0.940 ± 0.007	10.1 ± 1.0
d	0.91	0.832 ± 0.007	18.4 ± 0.9
e	0.88	0.886 ± 0.009	14.8 ± 1.2

Table 5: Linear fit on predictions in different cases.

In Figure 11, the linear fit of predicted ages is shown. Upon comparing[6] the R^2 values, slopes, and intercepts across different cases, we determined that superior results are obtained in case b.

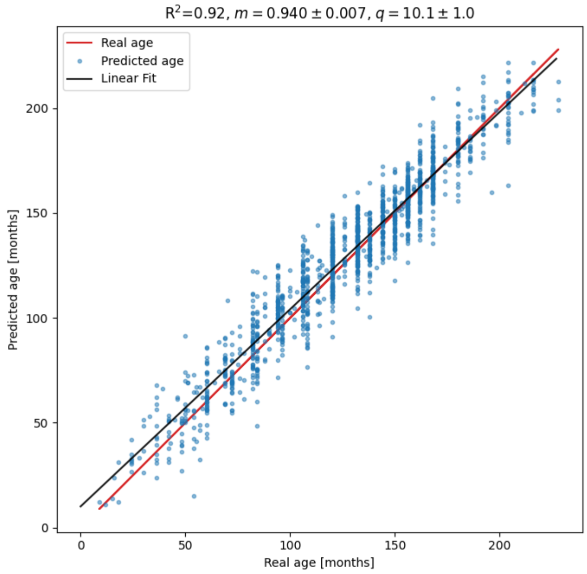

Figure 11: Linear fit of predicted ages in case b.

Hence, we can conclude that the optimal weights we identified are those presented in case b, making them the recommended choice for future predictions. Additionally, for quantifying error in future predictions, we have opted to utilize the Mean Absolute Deviation (MAD) from case b. With further testing, it's possible that even better results could be achieved.

[6] The Pearson coefficient, as it approaches 1, indicates a stronger linear relationship between the variables, similar to the criterion used for the slope. However, for the intercept, a smaller value is indicative of better training.

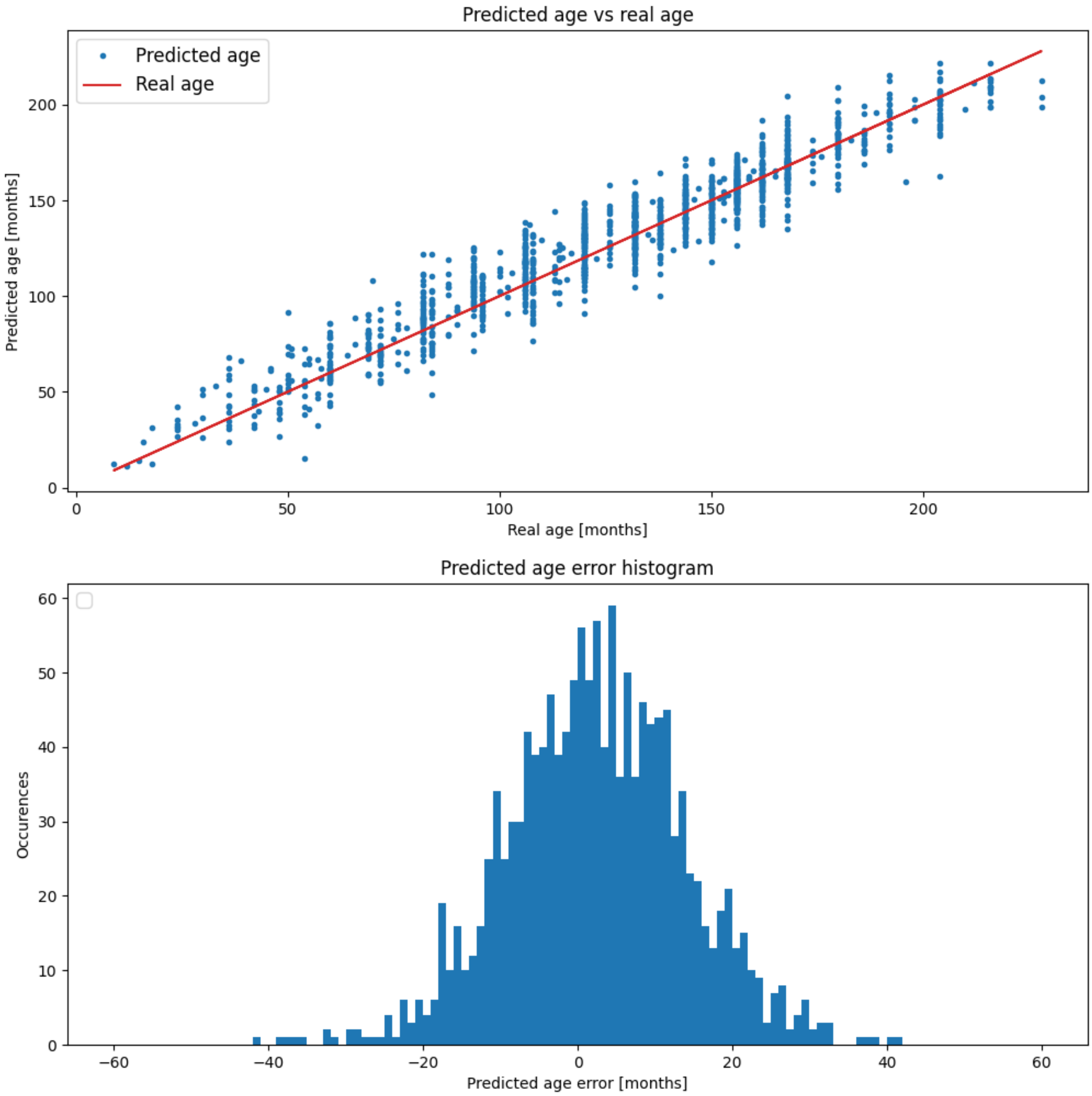

Figure 12: Output for model evaluation; at the top, a plot of predicted age versus real age reveals a fairly linear trend, indicating a good predictive capability of the model. At the bottom, a histogram illustrates the distribution of errors, calculated as the true value minus the predicted one. The distribution appears symmetrical and resembles a Gaussian curve, suggesting a balanced and accurate prediction performance.

4 Conclusions

The best result that we got from the model evaluation, a MAD of 9 months, is approximately twice as much the best MAD obtained in the RSNA challenge [5]. Therefore in this section we want to discuss the various reasons why our methods may be not as accurate as the ones presented in the challenge. Talking about the dataset we had available, we'd like to highlight some systematic issues that it owns. The radiography image quality is strongly dependent on a lot of factors, such as type of radiographer (whether it is digital or analogue); parameters of the scan that can affect contrast, brightness, field of view etc; artifacts produced by objects in the field of view, bad positioning artifacts, malformations and others. Thus if a standard scan procedure is not followed, every single image is different from the others in all of the aspects above. So the preprocessing step can become arbitrarily difficult without the appropriate methods and that was the case, because not all dataset images were taken with the same procedure or with the same instruments. We didn't use any specific hand segmentation techniques (that could have helped generalize the hand detection operation) because to train a network to detect hand shapes was a time and resource consuming task. In the RSNA challenge, some of the winners used these segmentation methods and got much better results in terms of MAD [4]. Moreover, our dataset was not balanced in terms of age and gender distribution. As for age distribution we tried an augmentation method that only consisted in random rotations of the images; the challenge participants also flipped the image, zoomed/unzoomed it and shifted it, but given our resources the rotation alone was a really demanding task. As for gender, we used a simplistic model that didn't take into account gender for bone age assessment; for example the first place winners of the challenge used a "gender network" that was concatenated to the "bone age network" and could predict the gender of the subject. This is a very important aspect, since sex plays a very important role in bone growth [6]. Finally, the custom VGG-16 network we utilized might not be optimal for this task due to its lack of explainability. To address this limitation, we propose incorporating an attention layer to identify the key regions of the hand where the network focuses its learning. Figure 14 presents a visual representation of the `rVGG16-L2` model with attention, created using the Python module `visualkeras` [2]. By integrating attention layers, as suggested by Chen et al. [1], it becomes feasible to utilize attention maps to pinpoint discriminative regions, as illustrated in Figure 13. This strategy holds promise for enhancing precision in the regression task.

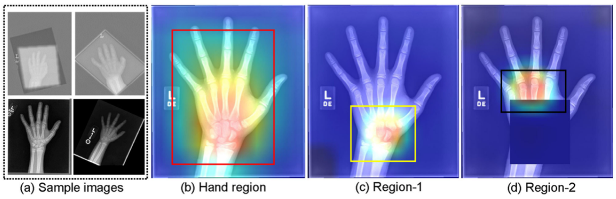

Figure 13: Displays RSNA bone age dataset images and corresponding attention heat maps. (a) Sample images. (b) Hand region attention map (H). (c) Most discriminative region (Region-1, R1) attention map. (d) Next discriminative region (Region-2, R2) attention map. [1]

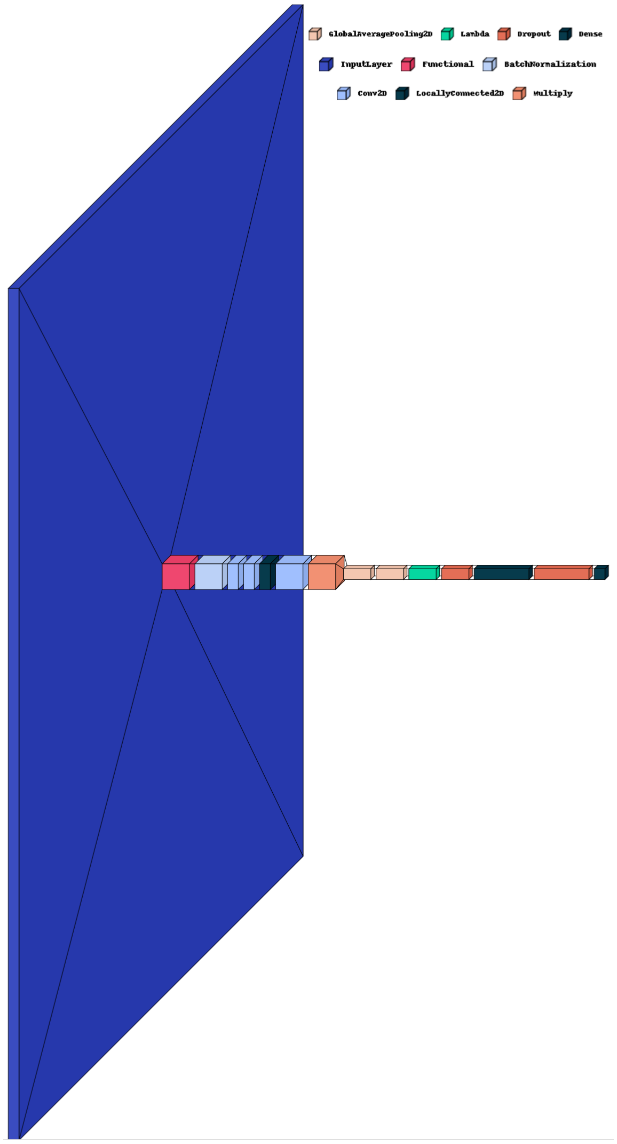

Figure 14: Illustration showcasing the integration of the rVGG16-L2 model with an attention mechanism. The VGG16 architecture is depicted within a Functional block in the visualization.

In conclusion, alternative models like Inception-V3 or ResNet-50 can be considered, although they typically require extensive training periods (about 50 hours for the first place Inception-V3 network). Based on this, exploring a transfer learning approach is viable, given its effectiveness in reducing training time.

References

[1] Chao Chen et al. "Attention-Guided Discriminative Region Localization and Label Distribution Learning for Bone Age Assessment". In: *IEEE Journal of Biomedical and Health Informatics* 26.3 (2022), pp. 1208–1218. DOI: 10.1109/JBHI.2021.3095128.

[2] Paul Gavrikov. *visualkeras*. https://github.com/paulgavrikov/visualkeras. 2020.

[3] Google. *Mediapipe | Google for Developers*. Last accessed 6 March 2024. URL: https://developers.google.com/mediapipe.

[4] Radiological Society of North America. *RSNA Pediatric Bone Age Challenge - Appendix (2017)*. Last accessed 6 March 2024. URL: https://bonexpert.com/refs/radiol.2018boneAgeChallengeAppendix.pdf.

[5] Radiological Society of North America. *RSNA Pediatric Bone Age Challenge (2017)*. Last accessed 6 March 2024. URL: https://www.rsna.org/rsnai/ai-image-challenge/RSNA-Pediatric-Bone-Age-Challenge-2017.

[6] Cole TJ et al. "Ethnic and sex differences in skeletal maturation among the Birth to Twenty cohort in South Africa". In: *Arch Dis Child. 2015 Feb* (2014). DOI: 10.1136/archdischild-2014-306399.

[7] Liu Z-Q et al. "Bone age recognition based on mask R-CNN using xception regression model". In: *Front. Physiol. 14:1062034.* (2023). DOI: 10.3389/fphys.2023.1062034.

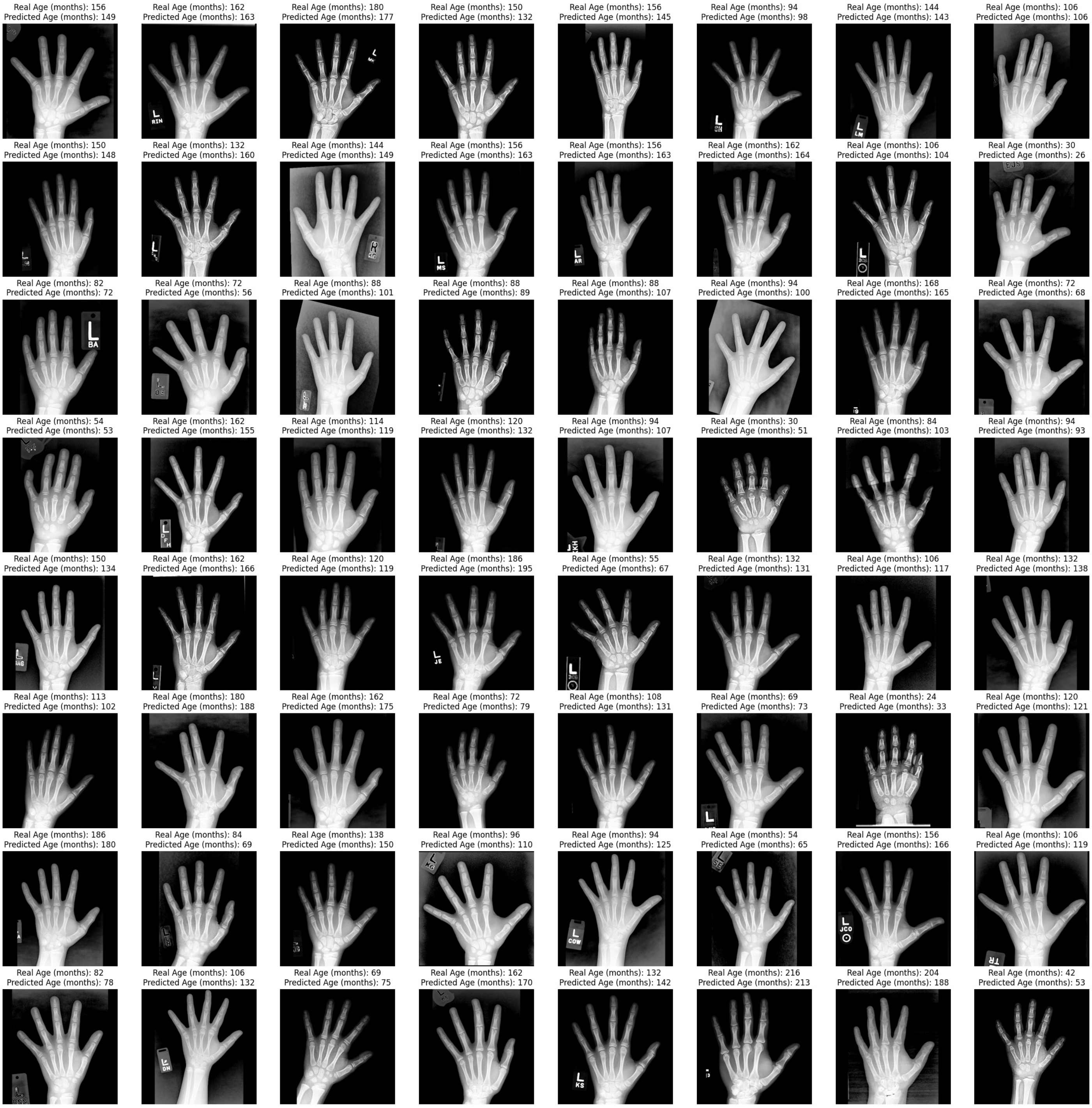
Real Age (months): 156
Predicted Age (months): 149
Real Age (months): 162
Predicted Age (months): 163
Real Age (months): 180
Predicted Age (months): 177
Real Age (months): 150
Predicted Age (months): 132
Real Age (months): 156
Predicted Age (months): 145
Real Age (months): 94
Predicted Age (months): 98
Real Age (months): 144
Predicted Age (months): 143
Real Age (months): 106
Predicted Age (months): 106
Real Age (months): 150
Predicted Age (months): 148
Real Age (months): 132
Predicted Age (months): 160
Real Age (months): 144
Predicted Age (months): 149
Real Age (months): 156
Predicted Age (months): 163
Real Age (months): 156
Predicted Age (months): 163
Real Age (months): 162
Predicted Age (months): 164
Real Age (months): 106
Predicted Age (months): 104
Real Age (months): 30
Predicted Age (months): 26
Real Age (months): 82
Predicted Age (months): 72
Real Age (months): 72
Predicted Age (months): 56
Real Age (months): 88
Predicted Age (months): 101
Real Age (months): 88
Predicted Age (months): 89
Real Age (months): 88
Predicted Age (months): 107
Real Age (months): 94
Predicted Age (months): 100
Real Age (months): 168
Predicted Age (months): 165
Real Age (months): 72
Predicted Age (months): 68
Real Age (months): 54
Predicted Age (months): 53
Real Age (months): 162
Predicted Age (months): 155
Real Age (months): 114
Predicted Age (months): 119
Real Age (months): 120
Predicted Age (months): 132
Real Age (months): 94
Predicted Age (months): 107
Real Age (months): 30
Predicted Age (months): 51
Real Age (months): 84
Predicted Age (months): 103
Real Age (months): 94
Predicted Age (months): 93
Real Age (months): 150
Predicted Age (months): 134
Real Age (months): 162
Predicted Age (months): 166
Real Age (months): 120
Predicted Age (months): 119
Real Age (months): 186
Predicted Age (months): 195
Real Age (months): 55
Predicted Age (months): 67
Real Age (months): 132
Predicted Age (months): 131
Real Age (months): 106
Predicted Age (months): 117
Real Age (months): 132
Predicted Age (months): 138
Real Age (months): 113
Predicted Age (months): 102
Real Age (months): 180
Predicted Age (months): 188
Real Age (months): 162
Predicted Age (months): 175
Real Age (months): 72
Predicted Age (months): 79
Real Age (months): 108
Predicted Age (months): 131
Real Age (months): 69
Predicted Age (months): 73
Real Age (months): 24
Predicted Age (months): 33
Real Age (months): 120
Predicted Age (months): 121
Real Age (months): 186
Predicted Age (months): 180
Real Age (months): 84
Predicted Age (months): 69
Real Age (months): 138
Predicted Age (months): 150
Real Age (months): 96
Predicted Age (months): 110
Real Age (months): 94
Predicted Age (months): 125
Real Age (months): 54
Predicted Age (months): 65
Real Age (months): 156
Predicted Age (months): 166
Real Age (months): 106
Predicted Age (months): 119
Real Age (months): 82
Predicted Age (months): 78
Real Age (months): 106
Predicted Age (months): 132
Real Age (months): 69
Predicted Age (months): 75
Real Age (months): 162
Predicted Age (months): 170
Real Age (months): 132
Predicted Age (months): 142
Real Age (months): 216
Predicted Age (months): 213
Real Age (months): 204
Predicted Age (months): 188
Real Age (months): 42
Predicted Age (months): 53

Figure 15: Here is an example of predicted ages made by the model. This is one of the outputs of the `model_evaluation` method of the `BoneAgeAssessment()` class.

Appendix A

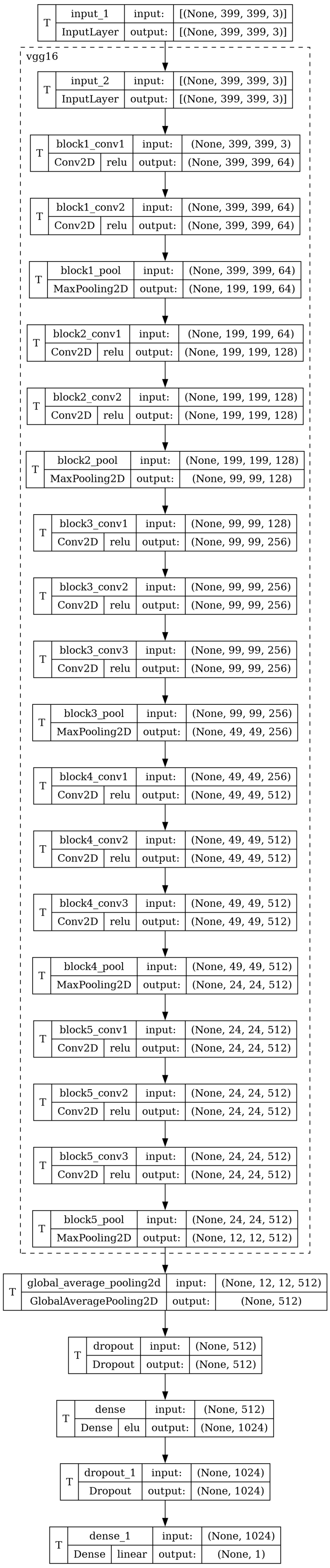
T | input_1 | input: | [(None, 399, 399, 3)]
InputLayer | output: | [(None, 399, 399, 3)]
vgg16
T | input_2 | input: | [(None, 399, 399, 3)]
InputLayer | output: | [(None, 399, 399, 3)]
T | block1_conv1 | input: | (None, 399, 399, 3)
Conv2D | relu | output: | (None, 399, 399, 64)
T | block1_conv2 | input: | (None, 399, 399, 64)
Conv2D | relu | output: | (None, 399, 399, 64)
T | block1_pool | input: | (None, 399, 399, 64)
MaxPooling2D | output: | (None, 199, 199, 64)
T | block2_conv1 | input: | (None, 199, 199, 64)
Conv2D | relu | output: | (None, 199, 199, 128)
T | block2_conv2 | input: | (None, 199, 199, 128)
Conv2D | relu | output: | (None, 199, 199, 128)
T | block2_pool | input: | (None, 199, 199, 128)
MaxPooling2D | output: | (None, 99, 99, 128)
T | block3_conv1 | input: | (None, 99, 99, 128)
Conv2D | relu | output: | (None, 99, 99, 256)
T | block3_conv2 | input: | (None, 99, 99, 256)
Conv2D | relu | output: | (None, 99, 99, 256)
T | block3_conv3 | input: | (None, 99, 99, 256)
Conv2D | relu | output: | (None, 99, 99, 256)
T | block3_pool | input: | (None, 99, 99, 256)
MaxPooling2D | output: | (None, 49, 49, 256)
T | block4_conv1 | input: | (None, 49, 49, 256)
Conv2D | relu | output: | (None, 49, 49, 512)
T | block4_conv2 | input: | (None, 49, 49, 512)
Conv2D | relu | output: | (None, 49, 49, 512)
T | block4_conv3 | input: | (None, 49, 49, 512)
Conv2D | relu | output: | (None, 49, 49, 512)
T | block4_pool | input: | (None, 49, 49, 512)
MaxPooling2D | output: | (None, 24, 24, 512)
T | block5_conv1 | input: | (None, 24, 24, 512)
Conv2D | relu | output: | (None, 24, 24, 512)
T | block5_conv2 | input: | (None, 24, 24, 512)
Conv2D | relu | output: | (None, 24, 24, 512)
T | block5_conv3 | input: | (None, 24, 24, 512)
Conv2D | relu | output: | (None, 24, 24, 512)
T | block5_pool | input: | (None, 24, 24, 512)
MaxPooling2D | output: | (None, 12, 12, 512)
T | global_average_pooling2d | input: | (None, 12, 12, 512)
GlobalAveragePooling2D | output: | (None, 512)
T | dropout | input: | (None, 512)
Dropout | output: | (None, 512)
T | dense | input: | (None, 512)
Dense | elu | output: | (None, 1024)
T | dropout_1 | input: | (None, 1024)
Dropout | output: | (None, 1024)
T | dense_1 | input: | (None, 1024)
Dense | linear | output: | (None, 1)

Figure 16: Full graph of `rVGG16-L2` model.