Bone Age Prediction Without Transfer Learning: A Study of ResNet18 with Attention and Inception-v4

Cesare Bidini†, Nicolò Rinaldi†

***Abstract*—Bone age prediction through automated analysis of radiographic images is essential for assessing pediatric growth to identify the presence of health issues or tracking the development of treatments. This study investigates the efficacy of training convolutional neural networks (CNNs) from scratch for bone age prediction using hand radiographs, a critical task in pediatric healthcare. Unlike state-of-the-art models that rely on transfer learning, we tested custom ResNet18 and Inception-v4 architectures to assess their predictive power. Additionally, we integrate Convolutional Block Attention Modules (CBAM) or Channel Attention Modules (CAM) to enhance feature extraction. Our approach includes extensive pre-processing and a targeted data augmentation strategy, focusing on the creation of 25 patches per image. Inference values are computed by extracting the median of the predicted values from patches for each image. Results demonstrate that Inception-v4 achieved a mean absolute error (MAE) of 6.33 months, outperforming the ResNet18 with CAM that has a MAE of 7,61 months. Nonetheless, ResNet18 remains a viable option in contexts where a model with lower space and time complexity is needed.**

***Index Terms*—CNN, Inception-v4, ResNet, CBAM, Channel attention, X-ray, Bone age prediction, Supervised Learning.**

I. Introduction

Bone age assessment is a standard procedure in pediatric and endocrinological settings. It is often used to evaluate children's growth path and to compare it with chronological age in order to determine the presence of health issues. For example, delayed bone age can be used to understand growth patterns and diagnose conditions such as growth hormone deficiency, hypothyroidism, malnutrition, and chronic illnesses. Furthermore, repeated bone age assessments can also monitor the efficacy of treatments over aforementioned diseases.

The most common method to perform the assessment is the radiography of the left hand with the wrist, because of the discriminant nature of bone ossification stages of the non-dominant hand (Giordano et al. [1]).

Bone age assessment is a natural task for machine and deep learning because it involves analyzing complex visual patterns in radiographic images, a process well-suited for algorithms that excel in image recognition and interpretation. Traditional methods for assessing bone age rely on expert interpretation, where a clinician visually compares X-ray images of a child's hand with standardized reference images. This requires deep domain knowledge, making it very prone to human error.

In this regard, to the best of our knowledge, methods that rely on transfer learning usually obtain better results (Rassmann et al., 2023 [2]). Transfer learning allows to leverage general purpose models pre-trained on big amount of data and fine tune them (or using them as feature extractor) on a downstream application. However transfer learning come with an issue: the primary training is commonly unrelated to medical tasks. Hence, the properties that the model learns may not be relevant to carry out medical tasks (Rassmann et al. [2]). In this paper we train a model from scratch in order to compare its results with previous works based transfer learning and assess the possibility of obtaining good results without resorting to it. Moreover, we test the performance of smaller models along with bigger ones to explore the possibility of obtaining reasonable results using a leaner and less deep architecture. In this regard we also test the role of attention as proposed by Woo et al. [3] using the entire CBAM module and only a part of it, to see if its introduction overperform the vanilla ResNet18.

For ease of use and simplified pre-processing of data and model integration, we develop a dedicated Python library, available on PyPI[1]. This library is imported and utilized directly within our notebooks to maintain a clean and organized workflow. The content of the library can be found in git-hub, at this link.

The present work is structured as follows: in Sec. II we provide a brief review of the literature about bone age prediction with CNNs, in Sec. III we describe the dataset and the pre-processing stages, alongside with the data augmentation strategy. The proposed architectures and training setup are outlined in Sec. IV, and the results are analyzed in Sec. V, followed by concluding remarks in Sec. VI.

II. Related Work

Traditional bone age assessment has historically relied on various methods, often yielding inconsistent results due to assessor variability. Over time, multiple automatic assessment methods have been explored, each with varying levels of success. The advent of fully automated assessment methods (Van Rijn et al. [4]), marked a significant advancement. In particular, applying deep learning algorithms to bone age prediction proved highly effective, especially with Convolutional Neural Networks (CNNs). Although CNNs were initially introduced by LeCunn et al. [5], they were applied to bone imaging data only relatively recently, establishing a new State-of-the-Art for bone age prediction on the public Digital Hand Atlas Database (Giordano et al. [1]). In their study, Giordano and colleagues tested three different approaches: using pre-trained CNNs as

†Department of Mathematics, University of Padova, email: {cesare.bidini, nicolo.rinaldi}@studenti.unipd.it

[1]https://pypi.org/project/Human-Data-Analytics/

feature extractors, fine-tuning these networks, or developing an ad hoc model called BoNet, which was trained from scratch for the task. Their findings revealed that BoNet performed best, with the most successful configurations not being those with the highest number of convolutional layers.

Later Larson et al., winner of the RSNA challenge, employed a ResNet-50 architecture with weights pretrained on ImageNet [6], which served as a baseline for the challenge. In the competition [7], Bilbily et al. and Pan et al. improved these results, achieving MAEs of 4.265 and 4.350 months, respectively.

Bilbily's approach used a pre-trained Inception-v3 model and incorporated gender information by concatenating the main models output with a 32-dimensional vector obtained through a dense layer. This combined representation was processed by additional dense layers to refine the embeddings. We decided to implement a similar structure to Bilbily's approach, but deploying the new version of the model, Inception-v4.

In contrast, Pan used two separate models for each gender, fine-tuning ResNet-50 models pre-trained on ImageNet. Their data augmentation strategy involved dividing each image into 49 overlapping patches, calculating predictions for each, and taking the Xth percentile of these predictions, where X was tuned on a validation set and averaged around 50 (median of the distribution). We decide to implement a lighter model then Pan, the so called ResNet18.

Following the success of transformer architecture (Vaswani et al. [8]), based on self-attention in Natural Language Processing, numerous attention-based CNN implementations have emerged in scientific research. Hassanin et al. [9] published a survey covering 70 of the most popular types of attention, highlighting that channel attention performs especially well for image classification tasks. Notably, they reference the Convolutional Block Attention Module (CBAM) by Woo et al. [3]. Inspired by their implementation, we incorporate this module into our ResNet-18 model, using both the full version (channel + spatial attention) and a reduced version (only channel attention), similarly to the approach taken by Liao et al. [10].

III. Data Preparation and Pipeline

The dataset, which is 10.3 GB in size, is composed of 14 236 clinical radiographs of the left hand of children in pediatric age coming from two institutions, Lucile Packard Children's Hospital at Stanford University and Children's Hospital in Colorado, then splitted in training, validation e test set following the distribution in Tab. 1. The images are stored as .png files with a unique integer name, used as index, and vary in dimensions. Therefore, we decided to standardize them as described in Sec. III-A.

For each split, there is an associated CSV/XLSX file containing three columns: index (int), male (bool), and bone age (float) in months. Since each split had a different file structure, we decided to standardize them by creating CSV files with consistent column names and deal with them as pandas DataFrame.

Data	Males	Famales	Total
Training set			
Stanford	1485	1200	2685
Colorado	5348	4578	9926
Total	6833	5778	12 611
Validation set			
Stanford	174	124	298
Colorado	599	528	1127
Total	773	652	1425
Test set			
Stanford	100	100	200

TABLE 1: Distribution of male and females in the RSNA dataset [7].

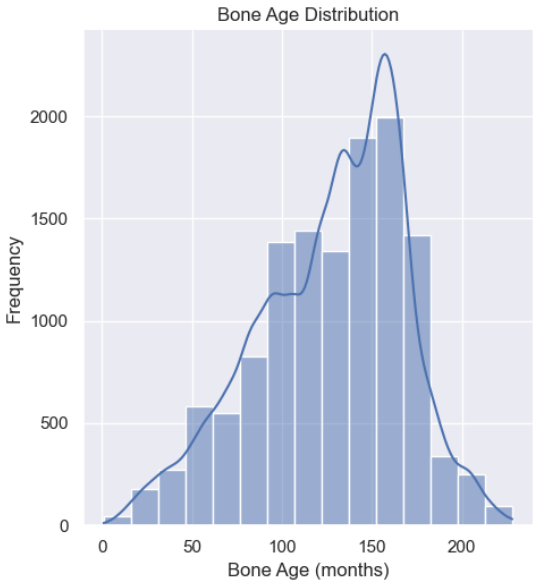

Fig. 1: Distribution of ages in the RSNA dataset [7].

A. Pre-processing

For pre-processing, we follow the standard approach adopted for X-ray images applied by almost all the participants, described in [7]. We have complied with the following steps:

1) **Crop:** We exclude manual cropping for obvious reasons of the size of the dataset being too big. We decide to slightly crop the images after a brief manual inspection revealed that they often included borders without any hand content.
2) **Resize:** Resizing is applied to standardize the image dimensions to 500×500 pixels.
3) **CLAHE:** CLAHE stands for Contrast Limited Adaptive Histogram Equalization. It is a method to enhance contrast in images that prevents over-amplification of noise in homogeneous areas. We apply it to each image.
4) **Normalization:** Each patch is normalized to obtain values in $[0, 1]$ for each channel.

The gender too is used as a feature. This information is

encoded in the dataset as a boolean value, we simply convert it to binary values: 1 for males and 0 for females.

An overview of the full pre-processing step can be seen in Fig. 2.

B. Data augmentation

After images pre-processing, we follow the approach described in [7] made by the second place Ian Pan, where each image is divided into n overlapping patches of size 224x224, where $n = 25$ was chosen taking into account computational limitations (Pan used $n = 49$). The positions of the 25 patches centers are determined sliding a window over the original image, with stride chosen in order to fit 5 patches for each side (Fig. 2). Multiple transformations are then randomly applied to the patches. The pool of possible transformations applied during training varies by model and is chosen through validation between horizontal/vertical flip, random rotation with $\alpha \in [-30^o, +30^o]$, gaussian blur with random $\sigma \in [0.2, 1]$, keeping obiouvly also the original patch. An example of possible augmentation is shown in Fig. 3.

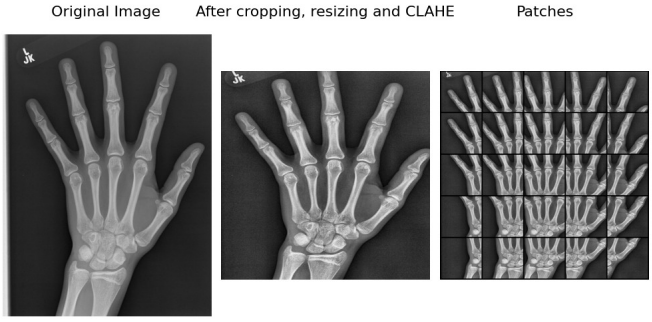

Fig. 2: Overview of the pre-processing step with 25 total patches.

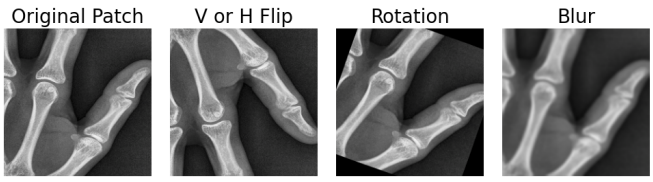

Fig. 3: Possible augmentations.

IV. Learning Framework

A. Architectures

1) ResNet18: The degradation problem highlights a limitation in training very deep networks effectively, where increasing depth paradoxically leads to poorer performance because current training methods fail to reach the best solutions as depth increases. He et al. [11] addressed this problem by adding shortcut connection and define a new building block for Convolutional Neural Networks: the residual block. The idea is to add a shortcut connection performing identity mapping and their outputs are added to the outputs of the stacked layers. The block structure can be more easily understood by looking at Fig. 4 (a).

Fig. 4: Taken from Liao et al. [10].

A potential solution for utilizing the aforementioned residual blocks, as proposed by He et al. [11], is the architecture known as ResNet18, which comprises a total of 17 convolutional layers and 1 fully connected layer. The overall structure is shown in Fig. 5. In this model, all the residual blocks are composed by two 2D convolution with kernel size (3,3), zero padding and the same number of filter depending on what stage. The first convolution is followed by the application of a ReLU activation function. If the first layer has stride (s,s) different from (1,1), the shapes are not compatible for an element-wise sum $\mathcal{F}(x) + x$ at the output of the residual block. This problem is avoided by applying a 2D convolution with kernel (1,1), stride (s,s) and same number of filters as the second convolutional layer of the residual block. After the tensor sum it is applied again the ReLU activation function. We will distinguish different stages of the model based on the colors shown in Fig. 5.

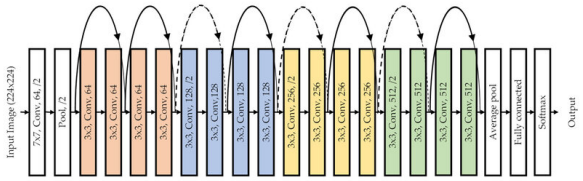

Fig. 5: Taken from He et al. [11].

As described in Sec. III, we will introduce as input inside the network RGB images with size 224×224, leading to a tensor of size $224 \times 224 \times 3$, always processed in batch.

Stage 1 (grey) This is only composed by a first convolutional layer with 64 filters a 7×7 filter with stride (2,2), activated through ReLU and then a 3×3 MaxPooling layerwith stride (2,2) is added, to be able to bring the images to the right dimension to be processed by the ResNet18 network.

Stage 2 (light blue) Now residual block are introduced. Both of them are equal with stride (1,1) and 64 filters each convolution. This is the only stage where the two blocks are exactly the same.

Stage 3 to 5 (green, pink and grey/blue) In this stages we have different residual blocks. The first with stride (2,2) and

the second with stride (1,1). The number of filters depends on the stage we are: respectively 128, 256, 512. At the end of the stage 5 the dimension of the output tensor is $7 \times 7 \times 512$.

Stage 6: Global Average 2D Pooling This stage is used to vectorize the output from stage 5, giving a vector of size 512. In this stage we also add a perceptron to integrate the gender information, similarly to first place Bilbily et al.(pag. 4 of RSNA 2017 challenge's appendix)[2] in [7] and a representation of size 16 is created. This is then concatenated to the initial 512-sized vector. Finally, we obtain a vector representation $\boldsymbol{v} \in \mathbb{R}^{528}$.

Stage 7 (yellow) At this step, the vector is processed by a perceptron layer with a weight matrix $\boldsymbol{W} \in \mathbb{R}^{528 \times n}$ where n is the number of classes for the classification/regression task, followed by the suitable activation function: sigmoid/softmax for classification or linear for regression. Since we are doing the latter, the weight matrix results to be $\boldsymbol{W} \in \mathbb{R}^{528 \times 1}$ and the activation function is the identity function.

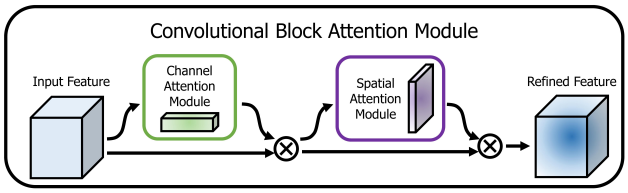

Fig. 6: CBAM module structure. Taken from Woo et al. [3].

2) Resnet with CBAM/CAM: Convolutional Block Attention Module (CBAM) is a self-attention mechanism, proposed by Woo et al. [3], designed to make use of both the channel and spatial dimensions. Directly computing a 3D attention tensor is computationally intensive, roughly doubling the overall processing cost. CBAM addresses this by decomposing the 3D attention tensor into 1D channel attention and 2D spatial attention, applying them sequentially to the input features to reduce computational load. This design, shown in Fig. 6, enhances efficiency. CBAM combines global average and global maximum pooling. These two statistics are supposed to complement each other, as using both has been shown (Woo et al. [3]) to outperform using either one alone. The channel and spatial attentions are sequentially applied to the input feature map F as follows:

$$SA(\boldsymbol{F}) = SA_{sp}(SA_{ch}(\boldsymbol{F})), \tag{1}$$

where SA_{ch} denotes the channel attention sub-module and SA_{sp} denotes the spatial attention sub-module which follows after the channel attention sub-module computation.

CAM: Channel attention module The structure of the channel attention module is illustrated in Fig. 7 (top). The attention tensor for the channel dimension is a 1D tensor. To efficiently calculate the 1D tensor, the global average and the global maximum values along each channel are pooled. After this, the 1D features are fed into a two-layer multi-layered perceptron activated with sigmoid function at the end. The mathematical notation of the channel attention calculation is:

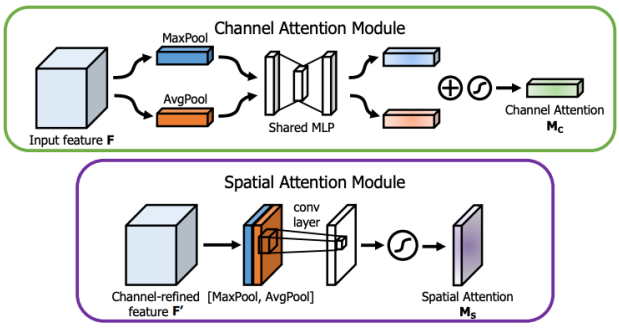

Fig. 7: Overall structure of the two building blocks of CBAM module: channel and spatial attention. Taken from Woo et al. [3].

$$\begin{aligned} \boldsymbol{F_{ch-avg}} &= \text{GlobalAveragePooling}(\boldsymbol{F}), \\ \boldsymbol{F_{ch-max}} &= \text{GlobalMaxPooling}(\boldsymbol{F}), \\ SA_{ch}(\boldsymbol{F}) &= \sigma(MLP(\boldsymbol{F_{ch-avg}}) + MLP(\boldsymbol{F_{ch-max}})) * \boldsymbol{F}, \end{aligned} \tag{2}$$

where σ denotes the sigmoid function, MLP is the two-layer multi-layered perceptron with two fully-connected layers activated by ReLU in between, $\boldsymbol{F_{ch-avg}}$ and $\boldsymbol{F_{ch-max}}$ are respectively global average pooled and global max pooled features along the channel dimension, where $\boldsymbol{F_{ch-avg}}$ and $\boldsymbol{F_{ch-max}} \in \mathbb{R}^{C \times 1 \times 1}$. The final output of the channel attention module is the original 3D CNN feature multiplied by the 1D attention tensor with broadcasting along the spatial dimension.

SAM: Spatial attention module The structure of the spatial attention module is illustrated in Fig. 7 (bottom). The mathematical notation for the spatial attention calculation is:

$$\begin{aligned} \boldsymbol{F_{ch}} &= SA_{ch}(\boldsymbol{F}), \\ \boldsymbol{F_{sp-avg}} &= \text{GlobalAveragePooling}(\boldsymbol{F_{ch}}), \\ \boldsymbol{F_{sp-max}} &= \text{GlobalMaxPooling}(\boldsymbol{F_{ch}}), \\ SA_{ap}(\boldsymbol{F_{ch}}) &= \sigma(Conv_{7\times7}([\boldsymbol{F_{sp-avg}}; \boldsymbol{F_{sp-max}}])) * \boldsymbol{F_{ch}}. \end{aligned} \tag{3}$$

Note that the input to the spatial attention module is the output from the channel attention module, $\boldsymbol{F_{ch}}$. As written in Eq. 3, the spatially avg/max pooled feature $\boldsymbol{F_{sp-avg}}$ $\boldsymbol{F_{sp-max}} \in \mathbb{R}^{1 \times H \times W}$ are fed into a convolutional layer to compute the spatial attention tensor of dimension $1 \times H \times W$.

Insertion of CBAM/CAM As cited by Woo et al. [3] and used by Liao et al. [10], a possible way to place the CBAM/CAM into ResNet-like architecture is to insert them after the second convolution layer inside the residual blocks.

The insertion position can be more easily understood by looking at Fig. 4 (b).

In this work we followed this approach and tested both a ResNet18 with CBAM module and one with only the CAM

[2]pag. 4 of RSNA appendix

module against a vanilla ResNet18, in order to assess the importance of attention and its parts.

3) Inception-v4: Inception-v1 (GoogLeNet), by Szegedy et al. [12], is a CNN based architecture whose main idea is the usage of inception modules: the modules output a vector of concatenated representations. The vector is obtained by the application of different kernel sized convolutions after a "bottleneck" of 1×1 convolutions on the output of the previous layer, that is used to reduce the number of parameters. This allows the original Inception-v1 module to capture characteristics that are distributed differently in the image while maintaining computational efficiency. Each iteration of the architecture built incrementally over the original one. Inception-v4, by the same authors [13], is the 4th generation of the architecture. It incorporates a greater number of inception modules, which contributes to improved performance in image classification tasks. An overall structure of the architecture is shown in Fig. 8.

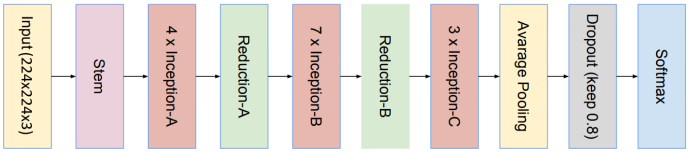

Fig. 8: Inception-v4 architecture. Taken from Szegedy et al. [13].

Key modules of Inception-v4 are:

Stem Module: an improved version of the inception-v1's stem module that perform more preliminary convolutions and contains the input part of the architecture.

Inception Modules: The model utilizes various types of inception modules (Inception-A, Inception-B, Inception-C) that allow for multi-scale feature extraction, effectively capturing different spatial hierarchies within the input data.

Reduction Modules: It employs reduction modules (Reduction-A and Reduction-B) to downsample feature maps while preserving important information, which helps in managing the computational load.

Batch Normalization This technique is integrated, already from Inception-v2, to stabilize learning and improve convergence speed by normalizing the inputs to each layer.

In our Inception-v4 model, as in ResNet18, we decide to add a dense layer with 16 neurons to integrate information from subject's gender, similarly to what was done by Bilbily et al. in [7]. The resulting vector is then concatenated to the output representation of the inception part (after the Dropout and before the last Dense linear layer used for regression) in order predict the bone age.

B. Training and inference setup

The training and inference setup for bone age prediction follows the same structure across both Inception-v4 and ResNet18 models, ensuring consistency in data handling and evaluation.

During training, each model receives the patches extracted from a dataset of preprocessed bone images. Specifically, 16k patches are randomly sampled from a fixed pool of possible patches and consequently randomly augmented. As described in Sec. III, we calculate the necessary stride and slid a 224×224 window across each image from the upper-left corner, generating patches to achieve the desired coverage with the goal of creating 25 patches. Each patch, once processed by the convolutional blocks of the model, is concatenated with the 32-dimensional vector output from a dense layer that take as input the gender information. This concatenated feature vector is then passed through a linear Dense layer to output a single prediction for the patch.

Along with learning rate, batch size, and loss function, crop size and augmentation types are considered hyper-parameters. Augmentation probability is uniform across four options, including no augmentation (identity), gaussian blur, random rotation and flipping, the latter being additionally divided in horizontal or vertical flip.

Early stopping monitoring validation loss is used as a method to select the best model, together with Adam as optimizer.

At inference, we compute the final prediction for each image by taking the median value of the predictions across its patches. This approach follows the method suggested by Pan et al. [7], but, in contrast, no additional tuning was conducted on the choice of the percentile to use.

Training posed significant challenges due to the dataset's size and architectural choices. TensorFlow's built-in generator functions struggled with our double-input structure and concatenation, despite thorough documentation research. Preprocessing and augmenting the dataset in advance was unfeasible, as cloud storage constraints and treating data augmentation as hyperparameters would have required multiple dataset uploads. We implemented a custom generator to dynamically augment images, avoiding excessive RAM usage. However, this solution introduced a CPU bottleneck that slowed training, even with batch pre-fetching. Attempts to resolve this, such as CPU parallelization, shifting preprocessing to a GPU-based layer, and other online techniques, were unsuccessful. Training was conducted on 'Kaggle' [3], that offers 30 hours of free usage weekly using two Nvidia Tesla T4 GPUs.

Tab. 2 with some of the tested configurations can be found in the appendix A.

V. Results

Models' performances are evaluated on a test set composed by 200 X-ray hands images, 100 belonging to males and 100 to females. As discussed in Sec. IV-B, for each architecture the best models are chosen based on their results on the validation set during training. At inference time we use as prediction for

[3]https://www.kaggle.com/

each image the median of the predicted age for the 25 patches belonging to that image.

We test multiple hyperparameters configurations for each architecture, performing a non exhaustive search over the parameters in Tab. 2 due to time and computational reasons; we are going to discuss the most relevant configurations in two separate sections: one for ResNet18 and its variants and one for Inception-v4. We also show the validation results in Tab. 5 along with training time and test time over the entirety of the splits; in Tab. 3 are displayed the space complexity of vanilla ResNet18, its two variants's and inception-v4's one.

Learning rate	0,0001	0,0005	0,0003	0,001
Augmentation	Flip	Blur	Rotation	
Batch size	36		40	
Loss	mse		mae	
Crop (T,B,L,R)	0.05, 0.05, 0.05, 0.05		0.1, 0.05, 0.1, 0.1	

TABLE 2: Table of possible hyperparameters values used for our search. — Crop: T = Top, B = Bottom, L = Left, R = Right

A. *ResNet-based models*

For the ResNet18 and its variants the best configuration resulted to be the channel attention one with MSE as loss, learning rate of 0.0005 and random flip as augmentations, performed for 50 epochs. Across multiple variants, a learning rate of 0.0005 yielded more consistent results during training: smaller lr seemed too slow while bigger ones showed more erratic behavior in some situations. Flips were the only augmentations applied to most of the ResNet-based model,given that, having a less complex architecture than the Inception-v4, they tended to show some signal that we interpreted as underfitting during the training if we applied more transformations. Our best models achieved a test MAE (using the median of patch predictions as prediction) of 7,61. It's worth noting that the CBAM variant didn't perform better than the channel attention one, even though we can see that results obtained with these models are not so far apart. For all three variants the best crop was the less aggressive one.

B. *Inceptionv4-based models*

Inception-v4 configuration with the best validation loss performed significantly better, as expected, reaching a MAE of 6.33 months when using the median of patches' predictions. The hyper-parameters used for this models included a learning rate of 0.0001, MAE as loss function and a mix of augmentations (random flip, rotation, and blur), performed for 75 epochs. We also applied a different crop during preprocessing over the one used for ResNet because this seemed more beneficial for this model after some trials. A particular phenomenon that we observed for the Inception-v4 but not for the ResNet18 was that, during training, in multiple epochs and across different models, the validation loss tended to oscillate around the training loss, often with values below the latter (Fig. 10). This was true in later epochs too, when the opposite is usually expected. We tried to apply a less strong regularization limiting the number of transformations applied to data, noting that the behavior was only slightly less pronounced, but did not change much. We are not sure of why this happens, but we speculate that multiple reasons might be in play:

- we might have regularized too much applying still too many transformation;
- TensorFlow documentation reports that this behavior might be caused by how the model switch to prediction mode at validation time, turning off batch normalization and drop-out. This might cause a lower error on the validation set. (keras documentation);
- validation loss is computed at the end of the epoch, if the is not over-fitting during that epoch then the loss is computed incrementally during training averaging batch with higher loss while at validation time the model has already received all the weights update for the epoch.(keras documentation).

Anyway, we couldn't investigate further to pinpoint the cause, given the GPU time limitations.

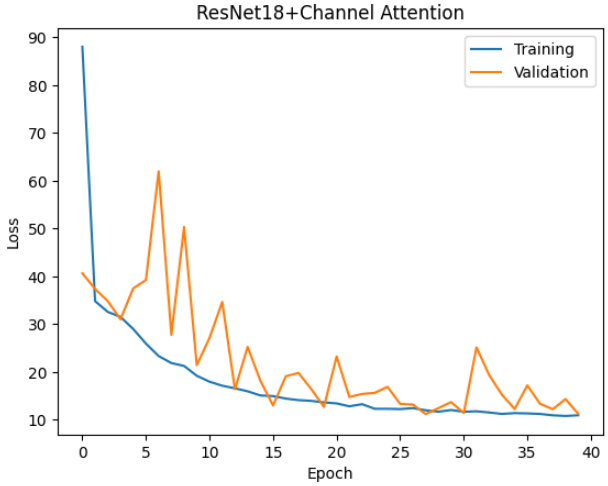

Fig. 9: Plot of train and val loss for ResNet18 with Channel attention module.

C. *Complexity*

Tab. 3 compares space complexity for the two main architectures and variants, with the amount of memory occupied by the model as displayed in Python, the size of the saved model's file and the number of parameters. Tab. 4 shows instead time complexity for four architecture both at training and at inference time. Despite the lengthy duration of the training for the models (in particular Inception) probably cause by what we supposed is a CPU bottleneck, this is anyway lifted at inference time, due to the absence of all the sampling and augmentations. This allows for a duration that should be more in line with what we expect for prediction's time on the 200 test's images. Inference on a single image for inception-v4 and ResNet18 CBAM, channel and vanilla is respectively

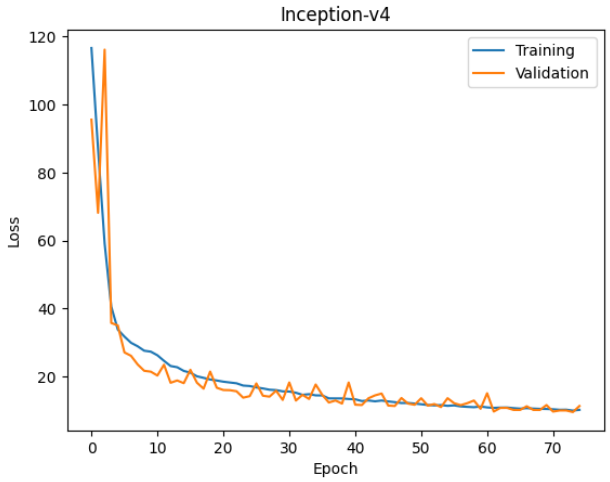

Fig. 10: Plot of train and val loss for Inception-v4.

366, 254, 249, 241 ms, following the same ordering as model size.

VI. Concluding Remarks

In this study, we focused on bone age prediction by reviewing and adapting relevant solutions from previous works. Unlike many state-of-the-art (SOTA) methods that utilize transfer learning, we trained models from scratch to evaluate the performance of non-pre-trained models. Our preprocessing involved enhancing and resizing images, with a tunable crop. We trained models on 16k randomly sampled patches per epoch from a pool composed by all the 25 patch groups of each image, applying random augmentations like flipping, rotation, and blurring. During inference, the median of the predictions for an image's 25 patches was used as final prediction to determine the bone age.

Our findings revealed that CBAM did not enhance the performance of channel attention ResNet18, showing worse results. The best models, ResNet18 with Channel Attention and Inception-v4, achieved mean absolute errors (MAE) of 7.61 and 6.33 on the test set, respectively. The use of median of patches' predictions reduced MAE with respect to validation by leveraging assessments from different image's regions.

Our ResNet-based model can be integrated into a multi-stage medical pipeline for automatic bone age assessment, especially in resource-limited settings with less powerful equipment. Critical cases with significant discrepancies between bone and chronological age can be further analyzed using the more robust Inception-v4 model and followed by a review from human experts.

A. Further Improvements

Future work should explore the many non tested hyper-parameter configurations. Given the singular behavior of the Inception-v4's training and validation curve (Sec. V-B), where we could notice the validation curve often being below the training curve, the further exploration of hyper-parameters might make the curves behaves as considered more common.

Testing different numbers of patches and samples per epoch could enhance performance, given that our choice was undertaken based on previous works and computational constraints.

Improving the cropping strategy to consistently include relevant image parts might reduce noise and improve training convergence. Additionally, optimizing the dynamic pre-processing pipeline could address the current training slow-down. Model selection could also be further inspected by using the median of the patches to compute validation loss, in place of computing the validation over each single patch.

B. Project's Considerations

The project was one of biggest we undertook from the side of space complexity and computational requirements; in this sense it made us better understand us how much of a trial and error process the training of these models is in contrast with our usual more theoretical approach. This also showed us the relevance of the optimization part and the importance of space and time constraints for a project feasibility. We also gained much more confidence in the use of TensorFlow, being mostly trained in PyTorch. The main difficulty we faced was related to both the construction of a custom pipeline and the optimization of it. In fact, we faced problems using the built in functions of Tensor Flow in association with our pipeline and managing the online pre-processing of this many data in an efficient way, having to resort to a solution that slowed down the training.

References

[1] C. Spampinato, S. Palazzo, D. Giordano, M. Aldinucci, and R. Leonardi, "Deep learning for automated skeletal bone age assessment in x-ray images," *Medical image analysis*, vol. 36, 02 2017.

[2] S. Rassmann, A. Keller, K. Skaf, A. Hustinx, R. Gausche, M. A. Ibarra-Arrelano, T.-C. Hsieh, Y. E. Madajieu, M. M. Nöthen, R. Pfäffle, *et al.*, "Deeplasia: deep learning for bone age assessment validated on skeletal dysplasias," *Pediatric Radiology*, pp. 1–14, 2023.

[3] S. Woo, J. Park, J.-Y. Lee, and I. S. Kweon, "Cbam: Convolutional block attention module," 2018.

[4] R. R. van Rijn and H. H. Thodberg, "Bone age assessment: automated techniques coming of age?," *Acta Radiologica*, vol. 54, no. 9, pp. 1024–1029, 2013.

[5] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[6] D. B. Larson, M.-C. Chen, M. P. Lungren, S. S. Halabi, N. V. Stence, and C. P. Langlotz, "Performance of a deep-learning neural network model in assessing skeletal maturity on pediatric hand radiographs," *Radiology*, vol. 287, no. 1, pp. 313–322, 2018.

[7] S. S. Halabi, L. M. Prevedello, J. Kalpathy-Cramer, and et al., "The rsna pediatric bone age machine learning challenge," *Radiology*, vol. 290, no. 2, pp. 498–503, 2019.

[8] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2023.

[9] M. Hassanin, S. Anwar, I. Radwan, F. S. Khan, and A. Mian, "Visual attention methods in deep learning: An in-depth survey," *Information Fusion*, vol. 108, p. 102417, 2024.

[10] J. Liao, L. Yuanchang, T. Ma, S. He, X. Liu, and G. He, "Facial expression recognition methods in the wild based on fusion feature of attention mechanism and lbp," *Sensors*, vol. 23, p. 4204, 04 2023.

[11] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015.

[12] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," 2014.

[13] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," 2016.

APPENDIX

Model	**Memory** (MB)	**File Size** (MB)	**Num Param**
ResNet18+CBAM	43.37	136.94	11,369,691
ResNet18+CAM	43.37	136.83	11,368,809
ResNet18(vanilla)	42.69	134.6	11,191,473
Inception-v4	157.31	496.56	41,239,153

TABLE 3: Space complexity of deployed architectures

Model	**LR**	**Val MAE**	**Test w/ Median**	**Loss**	**Augment**	**Crop (T,B,L,R)**	**Train** (s)	**Test** (s)
ResNet18+CBAM	0.0005	10.92	8.82	mse	F	0.05, 0.05, 0.05, 0.05	37120	13.74
ResNet18+Channel	0.0005	11.17	7.61	mse	F	0.05, 0.05, 0.05, 0.05	32293*	13.43
ResNet18 (vanilla)	0.0005	10.74	8.66	mse	F	0.05, 0.05, 0.05, 0.05	36621	14.58
Inception-v4	0.0001	9.56	6.33	mae	F, R, B	0.1, 0.05, 0.1, 0.1	65149	25.20

TABLE 4: Augmentation: F = Flip, R = Rotate, B = Blur — Crop: T = Top, B = Bottom, L = Left, R = Right
*: different training time due to the early stopping mechanism

Model	**LR**	**Loss**	**Augment**	**Batch size**	**Crop (T,B,L,R)**	**Val MAE**
ResNet18+CBAM	0.001	mse	F	40	0.05, 0.05, 0.05, 0.05	12.17
ResNet18+CBAM	0.0001	mse	F	40	0.05, 0.05, 0.05, 0.05	13.09
ResNet18+CBAM	0.0005	mse	F	40	0.05, 0.05, 0.05, 0.05	10.92
ResNet18+CAM	0.0005	mse	F	40	0.05, 0.05, 0.05, 0.05	11.17
ResNet18 (vanilla)	0,0005	mse	F	40	0.05, 0.05, 0.05, 0.05	10,74
Inception-v4	0.0001	mae	F, R, B	40	0.05, 0.05, 0.05, 0.05	9.98
Inception-v4	0.0001	mae	F, R, B	40	0.1, 0.05, 0.1, 0.1	9.56
Inception-v4	0.0003	mae	F, R, B	40	0.1, 0.05, 0.1, 0.1	10.16

TABLE 5:
Augmentation: F = Flip, R = Rotate, B = Blur, I = Identity — Crop: T = Top, B = Bottom, L = Left, R = Right