

Bone Age Determination

In television series *Bones*, forensic anthropologist, Dr. Temperance "Bones" Brennan's ability to predict a number of qualities of a person just by examining bone X-rays, captured my imagination. So when I came across a bones dataset in Kaggle, while deciding my final project, I instantly finalized it to apply my new skills in Deep Neural Networks.

Project Definition

Domain Background

Hand X-rays are performed for children to compare their skeletal age with actual age to identify any growth or hormonal abnormalities. Previously, determination of bone age was done by a visual evaluation of hand X-rays and manually comparing them to the Greulich and Pyle atlas.

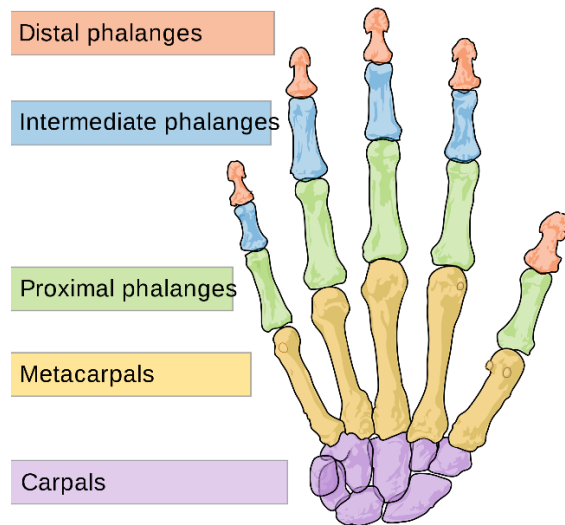


Figure 1: Human hand bones

Source: <https://goo.gl/images/SU4xxu>

There is an established order of ossification (bone tissue formation) for the carpal, metacarpal and phalangeal bones (Fig. 1), which is fairly constant and same for both genders. However, female bones mature faster than male bones. Vicente Gilsanz and Osman Ratib, in their book *Hand Bone Age* (http://www.chospab.es/biblioteca/DOCUMENTOS/Atlas_of_Hand_Bone_Age.pdf), have divided skeletal development in six categories, based on ossification of different bones of hands.

Category	Age Group (Males)	Age Group (Females)
Infancy	Birth to 14 months	Birth to 10 months
Toddlers	14 months to 3 years	10 months to 2 years
Pre-Puberty	3 to 9 years	2 to 7 years
Early and Mid-Puberty	9 to 14 years	7 to 13 years
Late Puberty	14 to 17 years	13 to 15 years
Post Puberty	17 to 19 years	15 to 17 years

Problem Statement

I plan to develop a machine learning model to determine skeletal age from a child's hand X-ray. This will be a regression problem where output will be a number in range of 1 to 230 for the skeletal age of the child. As we see from age ranges for categories of skeletal development, female skeletal matures earlier than male. So, apart from greyscale image of the hand X-ray, the child's gender will also be an input to the model.

Solution Statement

I will create the model using Convolutional Neural Networks to predict the bone age from X-ray of a child's hand. I aim to achieve a Mean Absolute Difference (MAD) of 8.0 or less for age.

I plan to code in Python and utilize the Keras library running on top of TensorFlow. After pre-processing the X-ray image, I'll use pre-built networks like Inception, DenseNet, etc. to extract its features. I will combine those features with the gender of the child and input it to 3-4 layers of fully connected layers. I will have one neuron in the output layer to predict the age.

Evaluation Metrics

The dataset that I am using is part of a challenge posted by Radiological Society of North America (RSNA) on Kaggle, as well as their website (http://rsnachallenges.cloudapp.net/competitions/4#learn_the_details-evaluation). The evaluation criteria mentioned by RSNA is MAD. Further, in my research I came across various Radiology and Oncology related papers that used MAD (Mean Absolute Distance/Difference) to compare different human reviewers or machine models.

Thus, I too will use MAD for evaluation of my model. MAD is a measure of statistical dispersion equal to the average absolute difference of two independent values drawn from a probability distribution. (https://en.wikipedia.org/wiki/Mean_absolute_difference)

Project Analysis

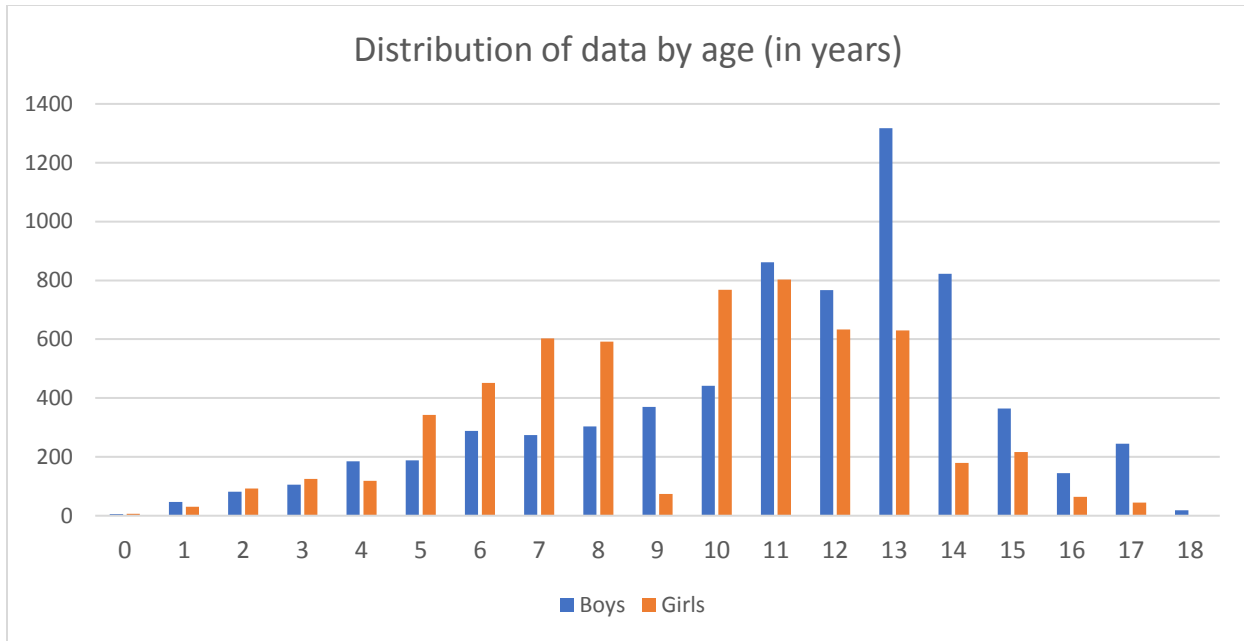
Datasets and Inputs

I will be using data made available by Radiological Society of North America (RSNA). It held a competition in 2017 to correctly identify the age of a child from an X-ray of their hand. The data is available in Kaggle at <https://www.kaggle.com/kmader/rsna-bone-age>

The data includes more than 12.5 k X-ray images which have been contributed by Stanford University, the University of Colorado and the University of California - Los Angeles.

Data Exploration

Of the 12,611 X-ray images in the dataset, ~54% are of boys (6,833) and rest are of girls (5,778). Below is histogram data for boys and girls age distribution with a bin size of 1 year.



The resolution of the images vary from 521 x 743 to 3001 x 2921. The file size varies from 98 kb to 3,940 kb. In some of the images, hand occupy most of the space, while in others it covers less than half the space. Further, the angles at which hands are places vary from image to image.

Below are the first 8 images in the dataset.



id: 1377



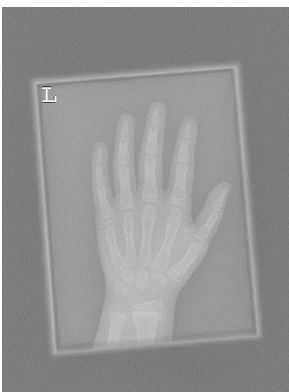
id: 1378



id: 1379



id: 1380



id: 1381



id: 1382



id: 1383



id: 1384

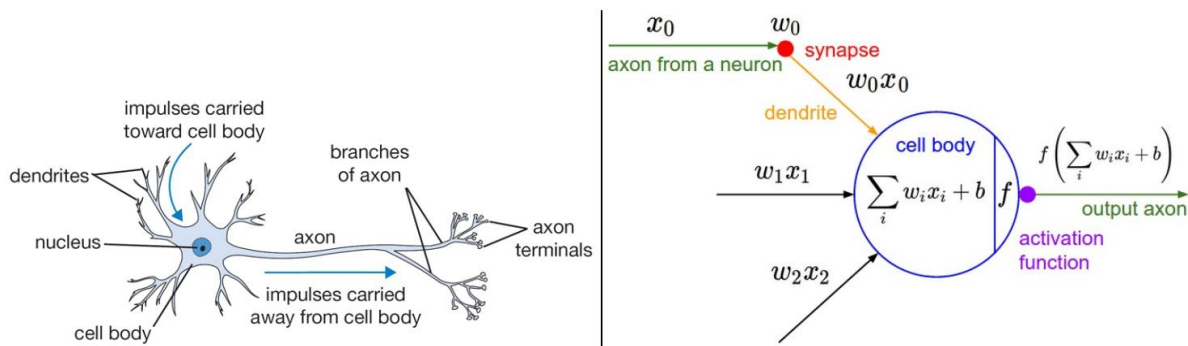
The corresponding genders and ages for the images above are:

ID	Gender	Age (in months)
1377	Female	180
1378	Female	12
1379	Female	94
1380	Male	120
1381	Female	82
1382	Male	138
1383	Male	150
1384	Male	156

Algorithms and Techniques

Neuron

Basic computational unit of brain is a neuron. Billions of neurons are present in human nervous system, which are connected through synapses. Each neuron receives input signals from its dendrites and produces an output signal along its axon. Below is a schematic diagram of a neuron (left) and equivalent computational model (right).

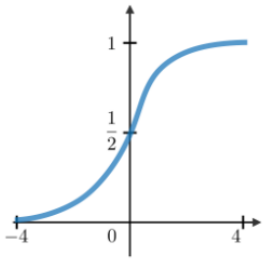
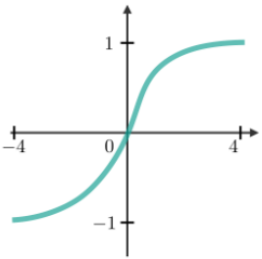
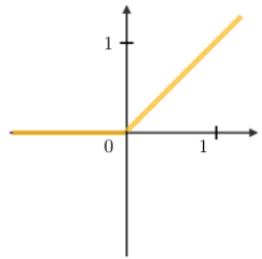
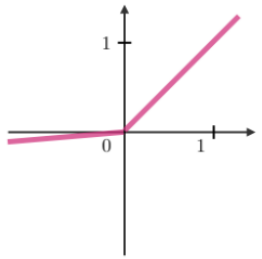


A cartoon drawing of a biological neuron (left) and its mathematical model (right).

Source: <http://cs231n.github.io/neural-networks-1/>

Activation Functions

When a neuron is 'fired' or activated is dependent on the inputs and the activation function. Below are some of the common activation functions used.

Sigmoid	Tanh	ReLU	Leaky ReLU
$g(z) = \frac{1}{1 + e^{-z}}$	$g(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$	$g(z) = \max(0, z)$	$g(z) = \max(\epsilon z, z)$ with $\epsilon \ll 1$
			

Source: <https://stanford.edu/~shervine/teaching/cs-229/cheatsheet-deep-learning.html>

A new activation function that has become popular nowadays is SeLU (scaled exponential linear units)

$$\text{selu}(x) = \lambda \begin{cases} x & \text{if } x > 0 \\ \alpha e^x - \alpha & \text{if } x \leq 0 \end{cases}$$

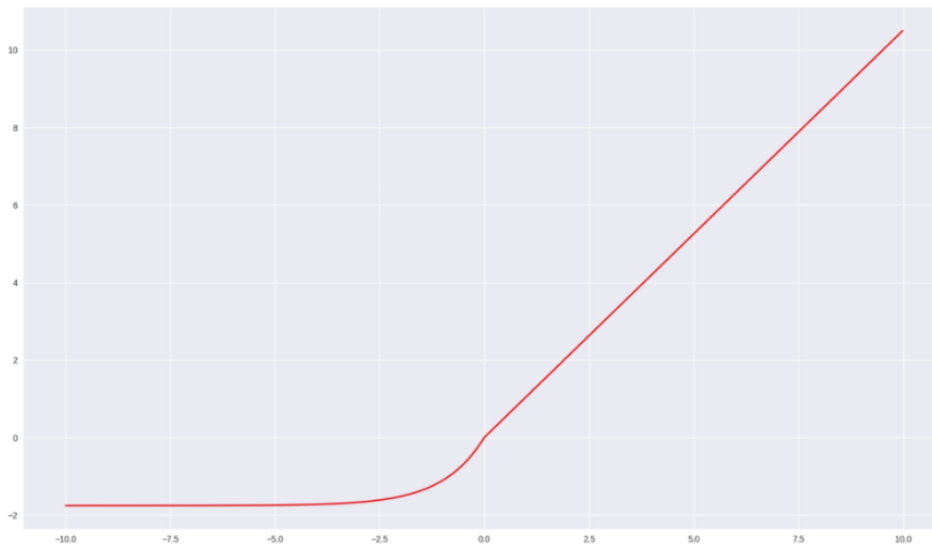


Figure 2 SELU plotted for $\alpha=1.6732\sim$, $\lambda=1.0507\sim$

Source: <https://towardsdatascience.com/selu-make-fnns-great-again-snn-8d61526802a9>

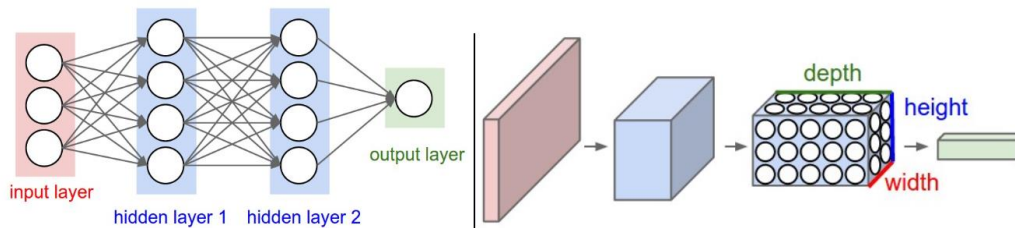
Neural Networks

Neural Networks are collections of neurons arranged in different layers. The outputs of some neurons are inputs to other neurons. Most common layer type is fully-connected layer in which neurons between two adjacent layers are completely connected pairwise, but neurons within a single layer do not have any connections.

Convolutional Neural Networks (CNN)

Convolutional Neural Networks are very similar to ordinary Neural Networks, they too are made up of neurons that have learnable weights and biases. However, to process an image by an ordinary neural network, an image has to be vectorized, i.e., two dimensional array of image data has to be converted to a one dimensional vector.

CNN architectures make an explicit assumption that the inputs are images thus making them much more efficient to process images and further vastly reduce the amount of parameters in the network.



Left: A regular 3-layer Neural Network. Right: A ConvNet arranges its neurons in three dimensions (width, height, depth), as visualized in one of the layers. Every layer of a ConvNet transforms the 3D input volume to a 3D output volume of neuron activations. In this example, the red input layer holds the image, so its width and height would be the dimensions of the image, and the depth would be 3 (Red, Green, Blue channels).

Source: <http://cs231n.github.io/convolutional-networks/>

Convolutional Layer

Conv layer is the core building block of a CNN. One of its parameters is set of learnable filter. Each filter is small spatially (2x2, 3x3, 4x4, etc.) but cover full channel depth (e.g., 3 for RGB image). Each filter is slid (or convolved) across the width and height of the image to produce a 2-dimensional activation map. The network learns filters that activate when they see some type of visual feature such as an edge of some orientation or a blotch of some color on the first layer, and eventually entire honeycomb or wheel-like patterns on higher layers of the network.

Pooling Layer

In between Conv layers in a CNN, usually a Pooling layer is inserted. It reduces spatial size of the layers and reduces amount of parameters for computation. Two common functions used for reducing parameters is MAX or AVERAGE, i.e., in a 2x2 or 3x3 window is reduced to one by selecting the max or average value in the window.

Fully-connected layer

Neurons in a fully connected layer have full connections to all neurons in the previous layer, just like a regular a Neural Network.

Transfer Learning

It is rare to train a CNN from scratch, rather pre-trained CNNs trained on huge datasets such as ImageNet with 1.2 million images and 10,000 categories are used as starting point. Either the pre-trained weights are used as initialization point, or features are extracted based on pretrained CNN and the extracted features are then used as input to a relatively less complex CNN.

Why CNN for this project?

As mentioned above, CNN makes an explicit assumption that input is an image, thus spatial information remain intact during analysis. For age determination, our model must be able to understand relationship between bones in different regions of the hand (carpal, meta-carpal, phalangeal – See Fig 1) Thus it makes sense to use a CNN for this project.

Benchmark Model

Mark Cicero and Alexander Bilbily model will be used as the benchmark model here. They achieved MAD of 4.265 on the test images and won the RSNA competition in 2017. Below is the schema of their final model.

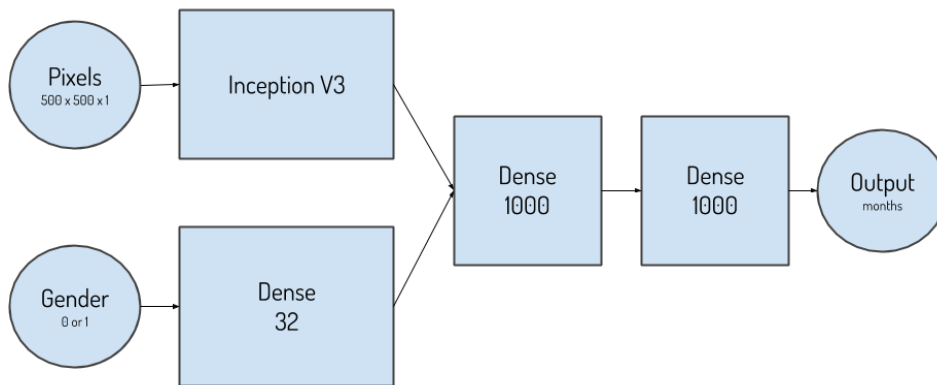


Figure 3: Schema of Mark Cicero and Alexander Bilbily's model

Source: <https://www.16bit.ai/blog/ml-and-future-of-radiology>

Project Methodology

Data Preprocessing

Creating Training, Validation, and Testing set

I divided the images data into training, validation, and testing, in the following ratio:

Training – 11,000

Validation – 1,400

Testing – 211.

Converting images to arrays of relevant sizes

I utilized keras preprocessing module to create arrays in a format consumable by built-in pre-trained models – Xception, ResNet50, and Inception_V3.

Feature Extraction and Saving NPZ files

I further extracted the features from the images by invoking the predict method of the pre-trained Xception, ResNet50, and Inception_V3 models.

Implementation

Language and Libraries

The coding is done in Python 3.6.4. I am using Anaconda navigator with Jupyter notebook. The following packages have been installed through Anaconda.

Package	Version	Description
scikit-learn	0.18.1	Set of python modules for machine learning and data mining
keras	2.0.2	Python Deep Learning library
numpy	1.13.3	Array processing for numbers, strings, records, and objects
pandas	0.23.4	Powerful python data analysis toolkit
tqdm	4.11.2	Fast, extensible progress meter
h5py	2.6.0	Pythonic interface to hdf5 binary data format

Generic CNN Model

I used function API of keras, to create two separate paths of the network:

1. Extracted features path
2. Gender path

I merged the two paths and added few fully connected layers followed by output layer with one neuron to predict the age.

Below is the general architecture that was used for this project.

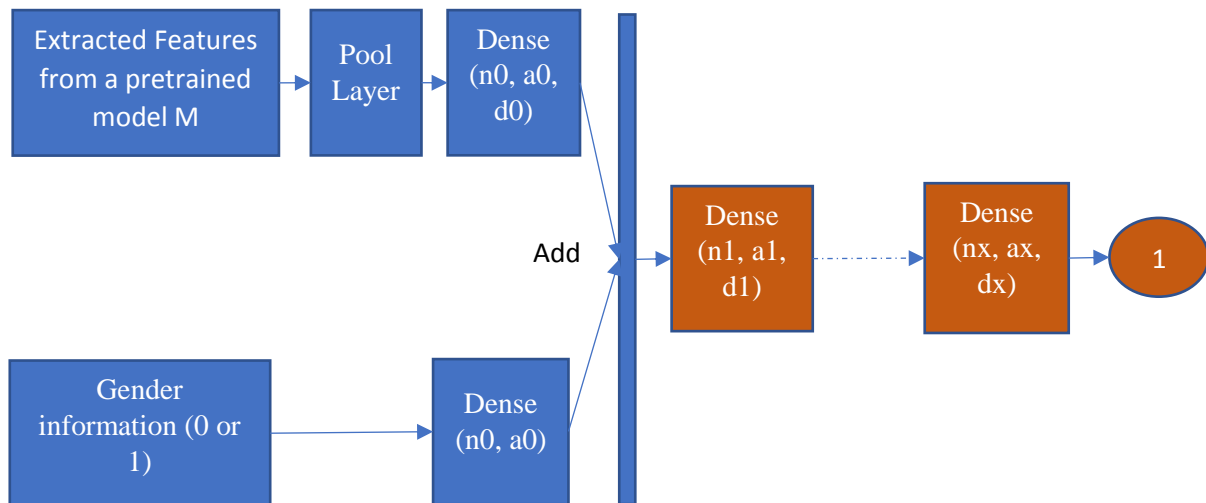


Figure 2: Schema of proposed CNN model

The table below lists the possible values of variables in the image above.

Pretrained model M	Xception, Inception V3, Resnet 50
Pool Layer	Max pooling + Flatten, Global Average Pooling
n0	32, 64, 128, etc.
a0	relu, selu

x (number of layers in orange part of network)	2, 3, 4
n1 ... nx	64, 128, 256, 512, 1032, etc.
a0 ... ax	relu, selu
d1 ... dx	dropout percent after the later

First I experimented with features extracted from different pre-trained models. I found Xception model to do the best. One issue I faced at this stage was having the incorrect input shape to receive the extracted features. As ResNet extracted features share is different than Xception and Inception V3.

M	Pooling	n0, a0,d0	x	(n1,a1,d1/b) ... (nx, ax,dx/b)	(Batchsize,Epochs)	MAD
Xception	Global Avg.	64, relu	2	(64,relu) (128, relu)	40,30	14.7
Xception	Global Avg.	32, relu	2	(64,relu) (128, relu)	40,30	12.6
Xception	Global Avg.	64, relu	3	(64,relu) (128, relu) (256, relu)	40,40	11.9
Inception	Global Avg.	64, relu	2	(64,relu) (128, relu)	40,30	12.88
Inception	Global Avg.	32, relu	2	(64,relu) (128, relu)	40,30	12.62
ResNet	Global Avg.	64, relu	2	(64,relu) (128, relu)	40,30	23.52
ResNet	Global Avg.	32, relu	2	(64,relu) (128, relu)	40,30	23.38

Then I tried a different activation function 'selu' instead of 'relu' and found it to be more effective. As 'selu' is a new addition to keras, I had to manually modify the activations.py file to get to work.

M	Pooling	n0, a0, d0	x	(n1,a1,d1/b) ... (nx, ax,dx/b)	(Batchsize,Epochs)	MAD
Xception	Global Avg.	64, selu	2	(64,selu) (128, selu)	40,30	11.9
Xception	Global Avg.	32,selu	3	(64,selu) (128, selu) (256, selu)	40,40	11.5

Then I experimented with different dropouts and number of neurons

M	Pooling	n0, a0, d0	x	(n1,a1,d1/b) ... (nx, ax,dx/b)	(Batchsize,Epochs)	MAD
Xception	Global Avg.	64, selu	3	(64, selu,d=.25) (128, selu, d=.25)(256,selu)	40,30	11.63
Xception	Global Avg.	32,selu	3	(64, selu,d=.25) (128, selu, d=.25)(256,selu)	40,30	11.88
Xception	Global Avg.	64, selu	3	(64, selu,d=.25) (128, selu, d=.25) (256,selu)	40,50	11.45
Xception	Global Avg.	128, selu, d=0.25	3	(128, selu,d=.25) (256, selu, d=.25) (128,selu)	40,50	11.06

Refinement

As can be seen above, I tried different options for pre-trained models, and number of neurons and activation functions. The best model achieved a MAD of 11.06.

Below is the summary of the final model.

FINAL PROJECT – UDACITY MACHINE LEARNING NANODEGREE

Layer (type)	Output Shape	Param #	Connected to
=====			
xception (InputLayer)	(None, 7, 7, 2048)	0	
=====			
global_average_pooling2d_59 (Glo	(None, 2048)	0	
dense_276 (Dense)	(None, 128)	262272	
gender (InputLayer)	(None, 1)	0	
dropout_79 (Dropout)	(None, 128)	0	
dense_277 (Dense)	(None, 128)	256	
add_59 (Add)	(None, 128)	0	
dense_278 (Dense)	(None, 128)	16512	
dropout_80 (Dropout)	(None, 128)	0	
dense_279 (Dense)	(None, 256)	33024	
dropout_81 (Dropout)	(None, 256)	0	
dense_280 (Dense)	(None, 128)	32896	
age_values (Dense)	(None, 1)	129	
=====			
Total params: 345,089.0			
Trainable params: 345,089.0			
Non-trainable params: 0.0			

Project Results

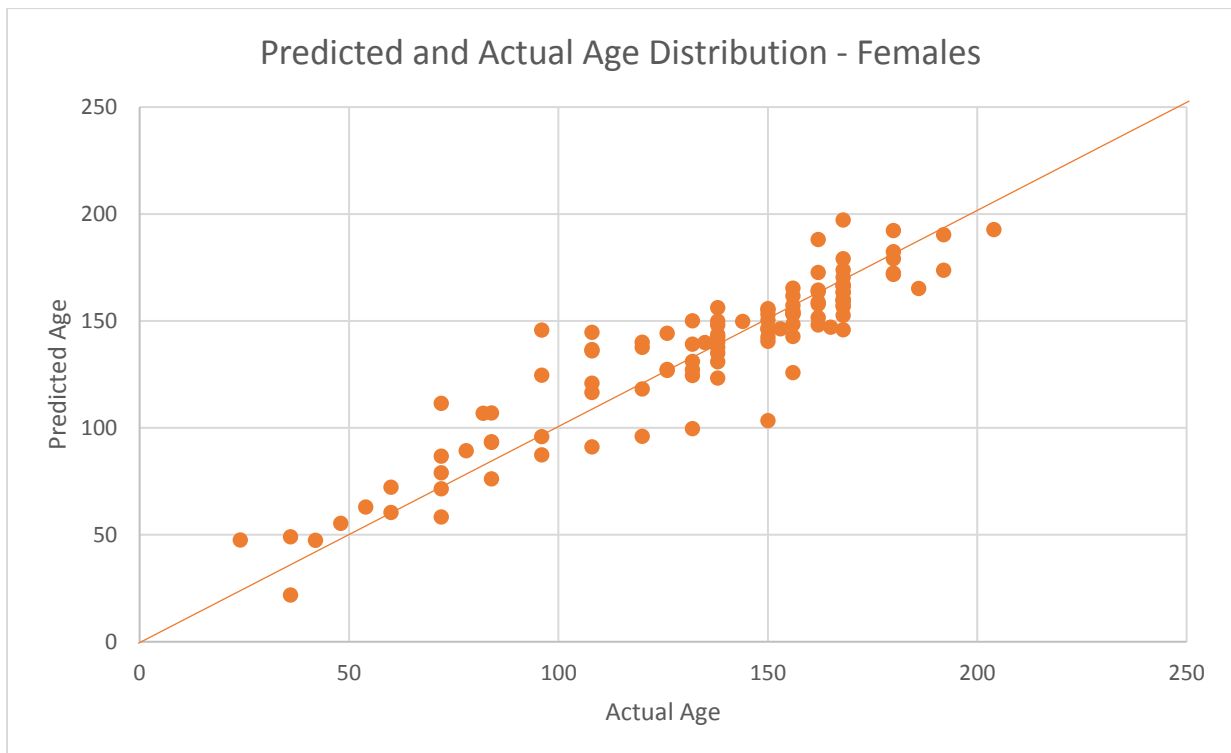
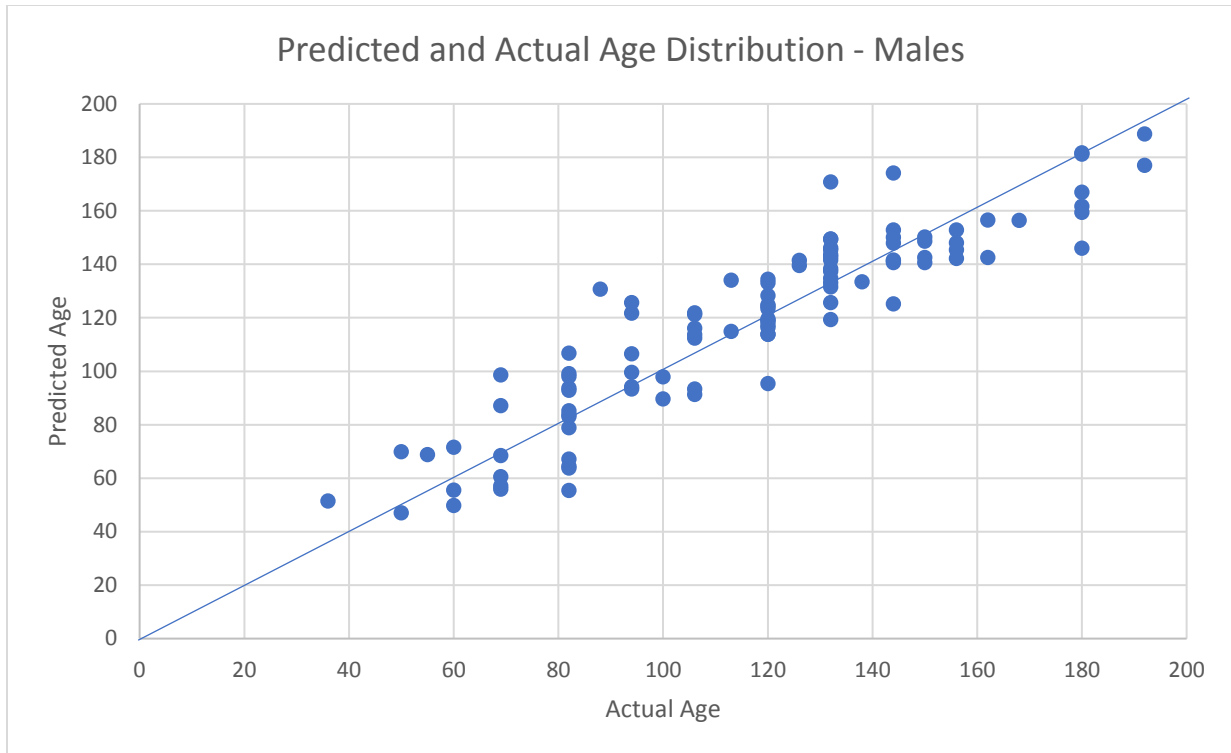
Model Evaluation and Validation

I started with an aim of reaching MAD of 8.0 or less. The best I could achieve is 11.06. Though it is higher than what I was initially aiming for, I am happy that my model is predicting ages with error less than a year (12 months)

Project Conclusion

Result Visualization

Below are two charts for predicted and actual age distributions – one for male and other for females. For both males and females, we see predicted age to be generally higher than actual age for ages less than 10 years (120 months), and generally lower for ages higher than 10 years.



Reflection

As can be seen from the charts above, for both males and females, the results are slightly higher for lower ages and slightly lesser for higher ages. It appears that the features model is extracting and relying upon to predict the ages are more prominent in smaller children and less in older age children.

Improvement – Future Work

I believe, one area that can significantly improve the MAD values is image augmentation. As a next project I will create additional images based on the images that are currently present in the dataset, and use increased dataset for training and validation.