EUROPEAN UNIVERSITY OF LEFKE

FACULTY OF ENGINEERING

Graduation Project 2

AGE ESTIMATION PROBLEM USING DEEP LEARNING MODELS

SALEM ABEDRABBO

20140004

Unleashing the Power of Deep Learning within the age detection , Us as people can sometimes guess and estimate people's age around us in my project am creating a convolution neural network - CNN deep learning model that can do this for us , motivation behind this project is to provide a highly precise tool for age regression , showing how strong the computer science technology and future is and how can developing a model can estimate an age of a human using the power of different deep learning pre-trained models .

Supervisor

VESILE EVRIM

Publish Date

Table Of Contents

1.Introduction

- ## 1.1 The age estimation problem

My focus is not on solving a traditional problem but on pushing the boundaries of what technologycan achieve in the realm of age estimation. This project doesn’t address a problem but rather than demonstrating the capability of the modern technologies through the lens of computer science , we will dive into the new technology and the art of the CNNs showing the power of deep learning models aiming to highlight the remarkable potential and advancement in this field. In this technological journey i will leverage CNNs into a peak of machine learning to dive into age detection , concentrating in features that these network forefront and the ability to automatically learn hierarchical representations from data which serves as the backbone of my journey. Features extraction that are taken out of face photos are essential for many uses A face image is a single point from which various attributes can be derived, including age, gender, race, and emotional state. If one could predict the age from people's facial images in real time, one would be able to control the content of the media that is read based on their age . Similarly, based on the customer's age and prediction, automatic prediction of age would help provide better buying preferences. Conventional machine learning techniques frequently include the manual definition of features, which may not always result in the best possible representations for the issues at hand. However, little processed input can be assumed by deep neural networks for image recognition, such as CNN, which then use a training process to determine the ideal network configuration. A unique kind of feed forward network called a convolutional neural network is mainly used to evaluate visual imagery. Convolutional neural networks are composed of neurons with learnable weights and biases, and they resemble ordinary neural networks in many ways. CNN distinct methodology allows them to outperform other deep neural network architectures in terms of performance. To comprehend an image, CNN aggregate several pixels together rather than examining each pixel separately.

1.2 Project Motivation and Objectives

The motivation behind this project is to create a highly precise tool for age regression, demonstrating the strength of computer science technology. Developing a model that can estimate the age of a human using various deep learning pre-trained models underscores the remarkable capabilities and advancements in computer science and comparing the results for each different pre-trained models as a starting point then taking the best model behaved with the dataset to the tuning the hyperparameters to compare the improvements .

Unleashing the strong capabilities of CNNs : creating a strong model in automatically learning hierarchical representation of the data.

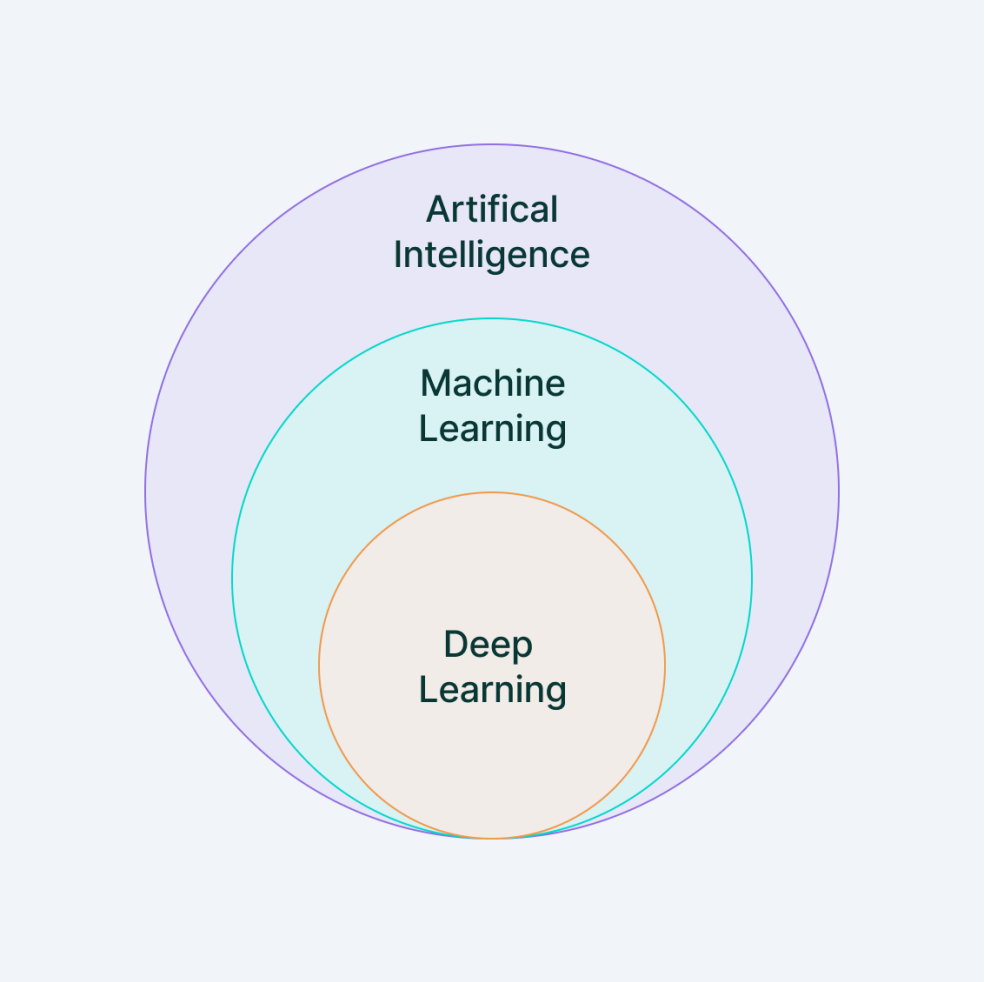
The **UTKFace** dataset contains around 23,000 images of human faces with different poses and lighting conditions. This makes it an ideal choice for the age estimation task. In recent years, there has been a lot of research in this field driven by the growing popularity of Deep Learning. Estimating age solely based on facial images is not easy, even with advanced Deep Learning techniques. There are many external factors that can affect someone's appearance and make age estimation difficult, such as their overall health, skincare routine, and genetics. Another challenge in this area has been the lack of high-quality labeled data to train deep models. However, this problem has been solved with the emergence of large labeled face datasets like UTKFace .
The idea of the Technology (Transfer Learning) which consists the use of models , that were already pre-trained on a very large datasets such as : ImageNet , is very useful when working with small datasets. I mainly focused on pushing kind of technology transfer learning using different pre-trained models where they were trained in ImageNet.

At the end after evaluating each models results I will save that model where I will be able to deploy it into an simple AI application on the test data.

Lets start by a question to get everything clear the technologies followed the courses went through to be able reaching my goal method.

What is machine learning ?

Machine learning (ML) is a subset from AI which exactly concentrates in data usage and algorithims where it's a computer based decision taking from a specific given data , same way how humans learn and gradually improves their leanings.

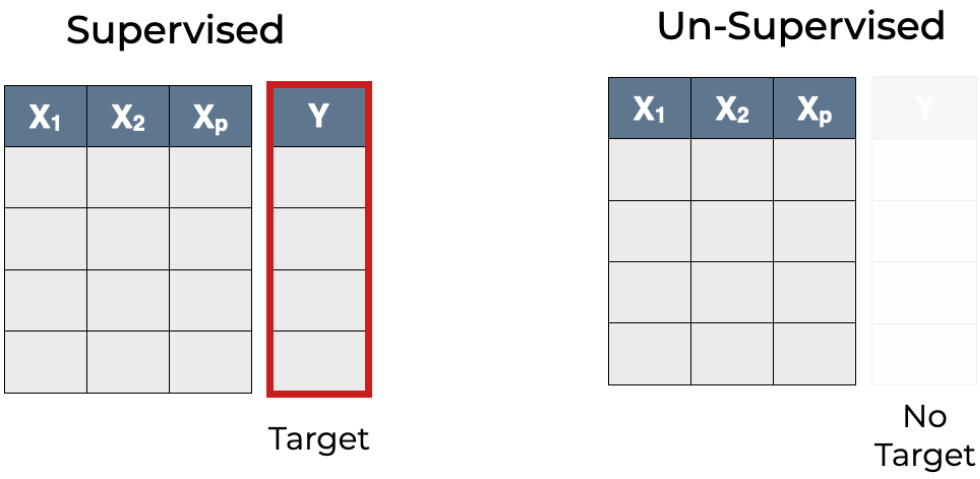


How ML works :

Decision process in general its used to do predictions based on some inputs data which sometimes can be labeled or unlabeled , ML algorithims will produce an estimated pattern in data in order to make predictions and decisions ,you can easily say it comes up with best guess function.

Machine learning methods :

Suprvised learning	**Unsuprvised learning**
● The data is labeled and you have the output of this data simply the x and y where x the input value and y is the target value. Which will lead to be able to predict to an unseen data .	● The goal is to infer the natural structure present within a set of data points.has no target value



In my case supervised learning where the UTKFace dataset has the labeled target value.

Where the process of ML involves as follows:

- **Training** : uses dataset to find patterns and relationships
- **Testing** : evaluating the model's behavior in the unseen data

second question approaches

What is Deep learning ?

Deep learning(DL) is subset of ML and an algorithm in ML that uses multi layer neural networks to make more complicated and complex procedures for data , its inspired from how human brain works we see and receive an input by eyes then this input goes to a neuron part of the brain, this neuron understands a specific part of an input then sends to another till its been collected to give an input.

Some of the supervised learning algorithms :

Neural Networks(NN): Models with interconnected neurons for capturing complex patterns where it has its own way learning patterns.

Where deep learning is a part of machine learning algorithm as we said it uses layers of neural networks to automatically discover the representations needed for the task directly from raw data.

1.5 Goals

- Improving the CNN base model for it's the unparalleled precision in such a project and dataset with a focus on improving overall accuracy , the diverse age annotations with the UTKFace.

- Integrating the age detection model into the real world applications and scenarios on the practical insights derived from the dataset deep range collection of the facial images.

- Evaluate and compare the performance of different CNN architectures in the realm of age detection, using the dataset I will choose as a benchmark.

- Investigate and contrast different methodologies employed in age detection, ranging from feature extraction techniques to training strategies, within the context of the dataset, Exploring how different CNN architectures and methods perform across the dataset, ensuring a comprehensive understanding of their adaptability pushing the boundaries of age prediction within the dataset.

- Predicting absolute age going through different CNN architectures and pre-trained architectures.

- Seeing the behavior of the dataset during training and try to do some techniques and tuning the hyperparameters to prevent overfitting.

- Pushing the transfer learning into the process and comparing different pre-trained architectures.

- Taking the best result from the pre-trained model and add some regularization techniques and reduce the overfitting.

- Defending my methodology and comparing it with related works.

2. Literature Survey

Compare1 :

Introduction

This is because age assessment based on facial images is a difficult task, which has great potential for the use in security systems, biometrics, and the interaction between people and computers. Most of the earlier approaches were primarily based on hand-engineered features and machine learning techniques and suffered from issues with accuracy and model generalization. However, the deep learning, especially the Convolutional Neural Networks (CNNs) has brought a tremendous change in this field by designating the data mining process to discover features automatically for better performance.

Early Methods

The trivial study on age estimation began using hand engineered features such as wrinkles, skin texture and geometry of the face, and training models of regression/ classification. However, these methods were very disadvantaged because they required the use of manually designed features, which in most cases did not capture the real changes in facial aging.

Deep Learning Advancements

The change of focus towards deep learning for more advanced methods has resulted in a positive impact on age estimation. The quest for complex representations and the scalability of architecture has made CNNs the center layer of most age estimation models modeled today. Famous networks as ResNet, Inception and DenseNet have been successfully used.

Transfer Learning

Fine-tuning a pre-trained deep learning model on a specific task is referred to as transfer learning and has been very essential when it comes to age estimation. As researchers suggested using models trained on large data sets such as ImageNet even limited amount of age-labeled data researchers are getting better results.

So I followed some method to compare it with this published paper[1] where I took a couple of training with different transfer learning architectures and added a couple of dense layers and regularization techniques in order to prevent the overfitting then take the best model behaved with the UTKFace dataset and take it into a tune hyperparameters in order to see what can I improve in the MAE (mean absolute error) result and overfit prevention. This paper used similar way of reducing the mae score pushing the transfer learning into work. And it did both classification and regression task for each pre-trained model , they took 3 classification tasks for each of individual age class,5 years age grouping classes and 10 years grouping classes and a Regression task for each .

Compare 2 :

this paper followed a different way of pre-processing and feature extracted ways then they can add it to deep CNN (DCNN). They converted the images into grayscale then histogram equalization to improve the contrast of the image histogram equalization is implemented. It uniformly distributes the pixel value of the image . Then a gabor kernel to detect edges . they used feature extraction using PCA .

Main motivation behind using PCA is, it consider top feature for dimensionality reduction.[2]

3. Background Information

3.1 Required & Used software

- **Spyder IDE :**

open-source cross-platform integrated development for doing multiple machine learning tasks since I have a low system RAM and low GPU RAMS moved to clouded enviroment which they provide high system RAMs and high GPUs.

- **TensorFlow :**

  A library in production of deep learning models.

- **Keras:**

  A high built in API built in tensorflow makes a strong user-friendly quickly building and testing networks.

3.2 Other software

- **Matplotlib :**

  library in python which will show us the visual presentation.

- **NumPy:**

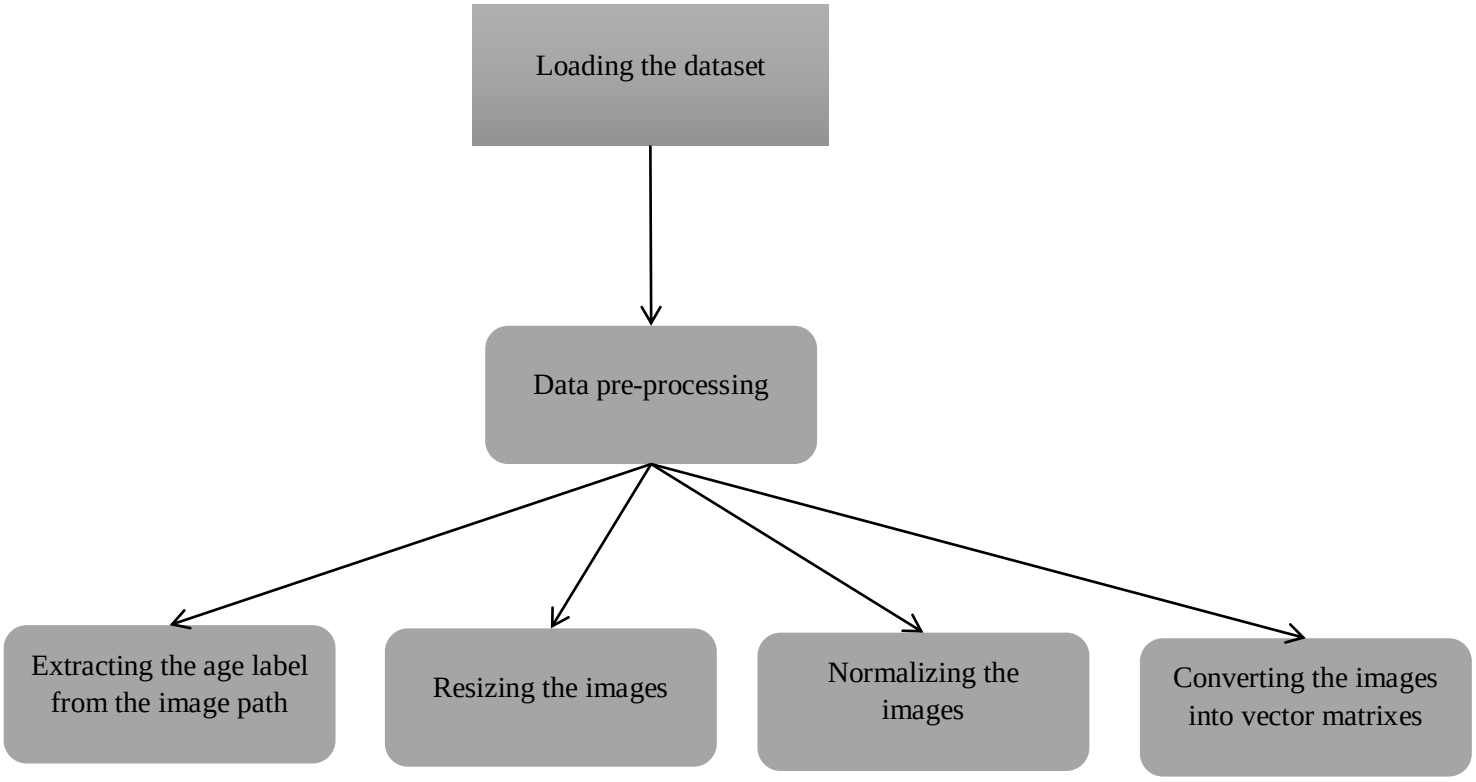
  Used for converting the images into a vector matrixes.

Google Colab Needed for high CPU and GPU

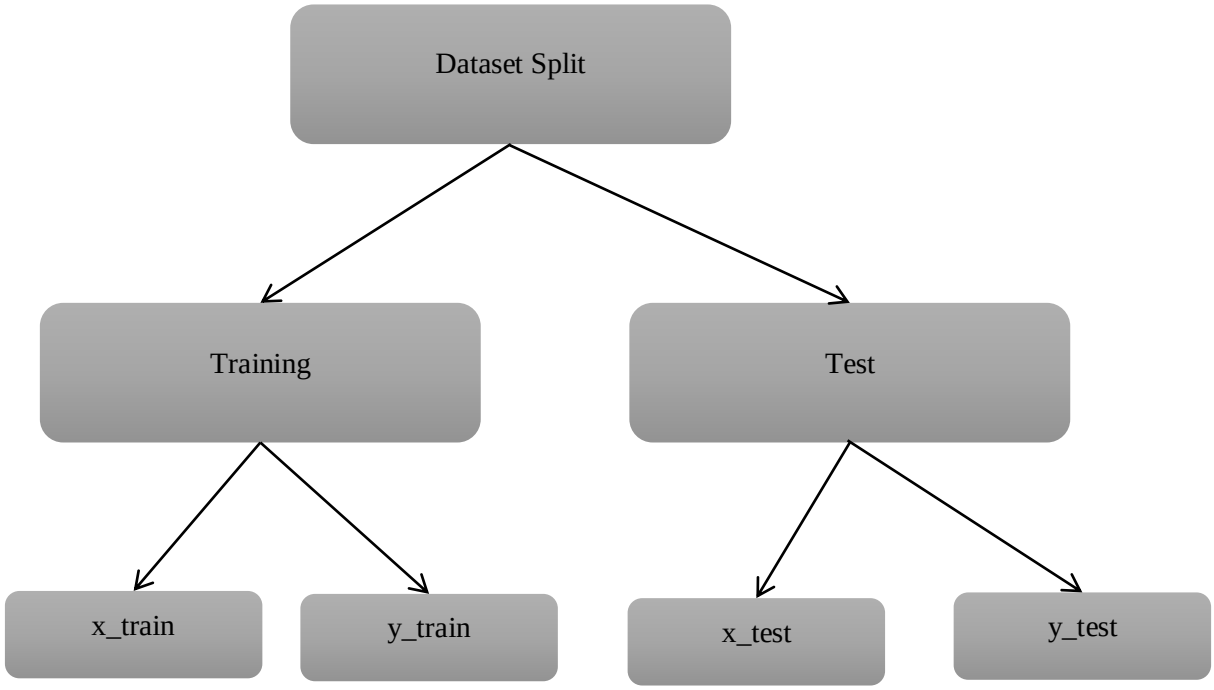
is an online service that lets you write and execute Python code in your browser, with access to GPUs and TPUs where I purchased Google Colab pro giving me the access into high GPU's and CPu since high-performance GPUs and CPUs are essential for age estimation tasks to handle the computational demands of model training, data processing, and experimentation efficiently.

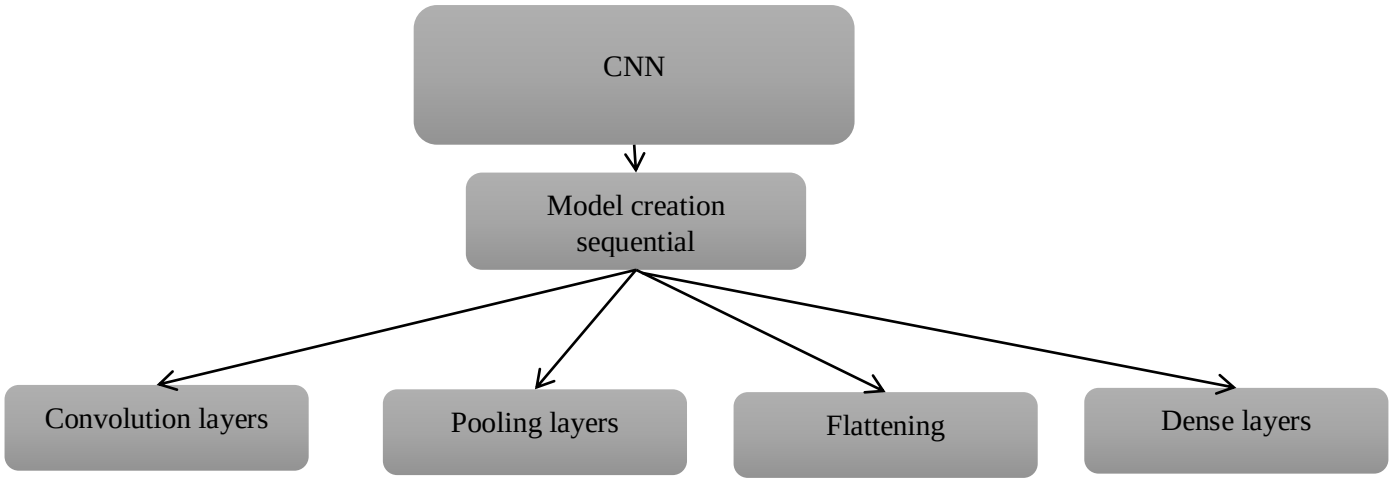
4. Design Documents

4.1 Data flow diagram

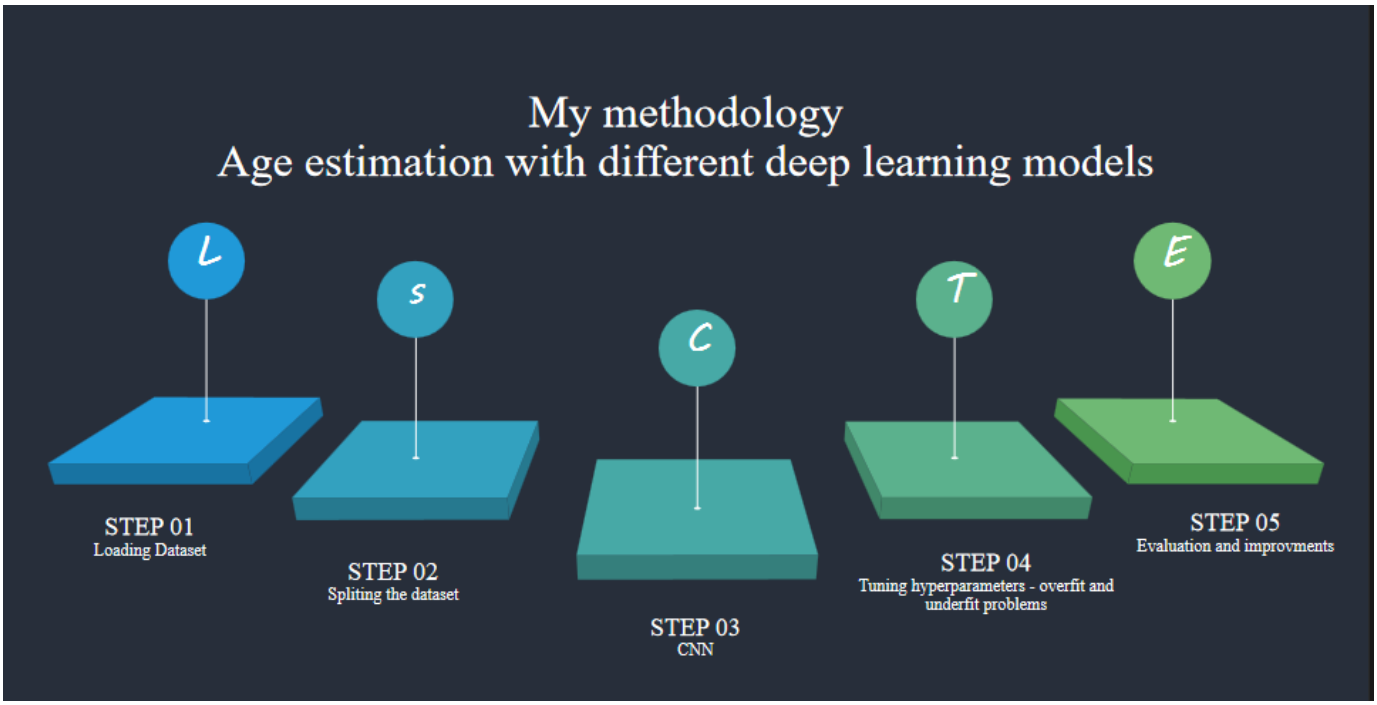


4.2 Data flow diagram





5. Methodology



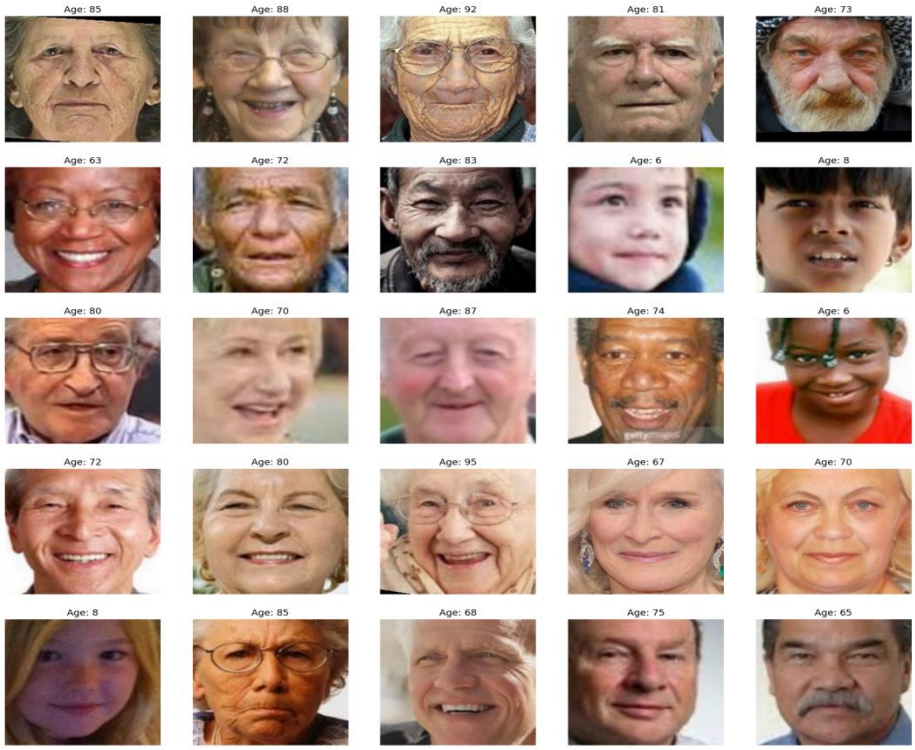
Loading the Dataset

The **UTKFace** dataset contains around 23,000 images of human faces with different poses and lighting conditions. This makes it an ideal choice for the age estimation task. In recent years, there has been a lot of research in this field driven by the growing popularity of Deep Learning. Estimating age solely based on facial images is not easy, even with advanced Deep Learning techniques. There are many external factors that can affect someone's appearance and make age estimation difficult, such as their overall health, skincare routine, and genetics. Another challenge in this area has been the lack of high-quality labeled data to train deep models. However, this problem has been solved with the emergence of large labeled face datasets like UTKFace .

The idea of the Technology (Transfer Learning) which consists the use of models , that were already pre-trained on a very large datasets such as : ImageNet , is very useful when working with small datasets. I mainly focused on pushing kind of technology transfer learning using different pre-trained models where they were trained in ImageNet.

This dataset comprises of 23,708 images of facial image indivisuals with an age range 0 - 116 with annoted age , gender , ethnicity where it consists of 52% males and 48% females where they are mostly balanced , where the age label is what we need from this dataset.

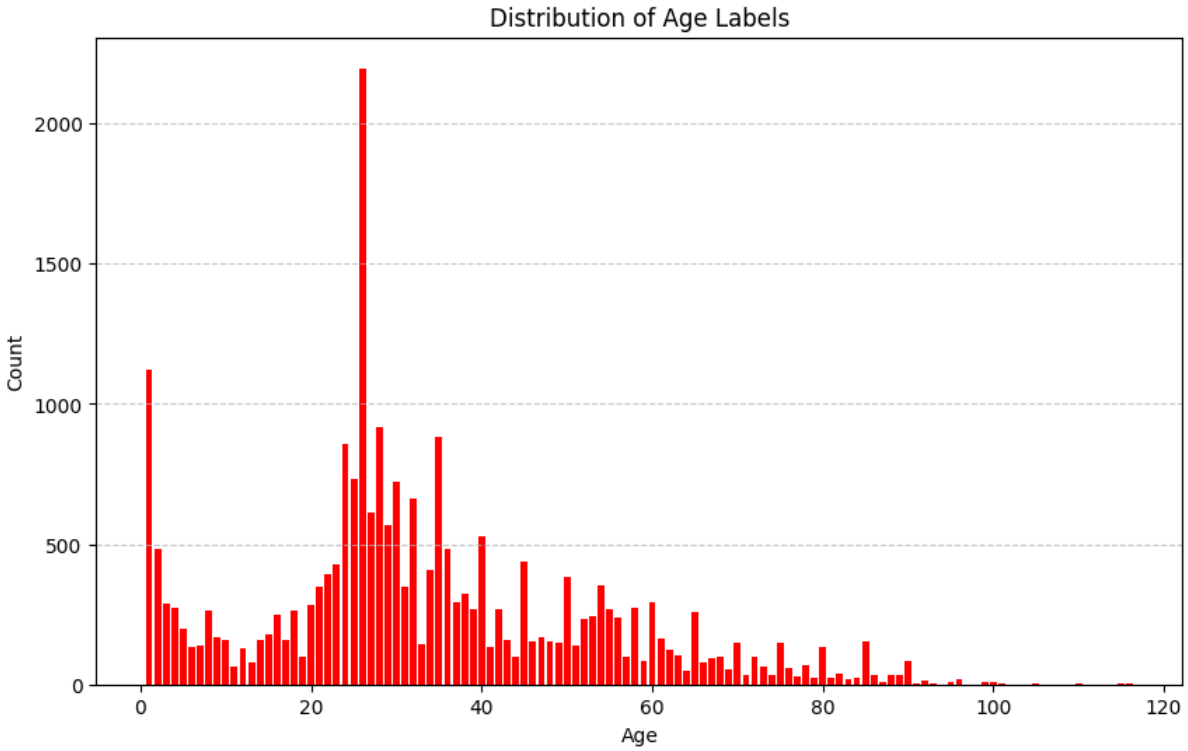
Samples with the corresponding age from the dataset plotted by python library in my notebook:



Age distribution :

To get more into understanding and examining the dataset I will show the age distribution where this is a very important aspect for me beacause it directly influences the data exploration I will go through wheither to understand the model and training behaviors.

This is the age distribution plotted from my notebook :



As we can see, there are quite a lot images of persons with ages 1 and ages between 20 and 40 compared to other age ranges.

Pre-processing steps

How the image looks and how the labels of the image is represented in each image :

The size of each images is 200 x 200

Image label represents first index as the Age label then second index is for the gender and third is for ethnicity and the rest is images ID .

25_0_1_20170113151453752.jpg.chip.jpg

Index 0 (age) : 25

Step 1 :

we will use a function to load the images from a specific folder .

Step 2 :

Initializing an 2 empty arrays to store the image with their corresponding age

Step 3:

An iteration process extracting the age information from the file names at the (_) by a delimiter

Resizing the images and Normalizing the images

- I resized the images into 150 x 150 as a starting point to see results

- Converting the images into vector matrices using the NumPy array

- Normalization for the images pixel value to be in a range between 0 - 1

The image size was 200 x 200 and it was reduced to 150 x 150 size as since I have a restricted computational resources where even after using the colab Pro.

Converting the images into a vector metrices so the system can understand a representative data.

Normalizing the images where all images pixel is in a range between 0 - 255 so normalizing pixel values to in a range between 0 - 1

Spliting the dataset

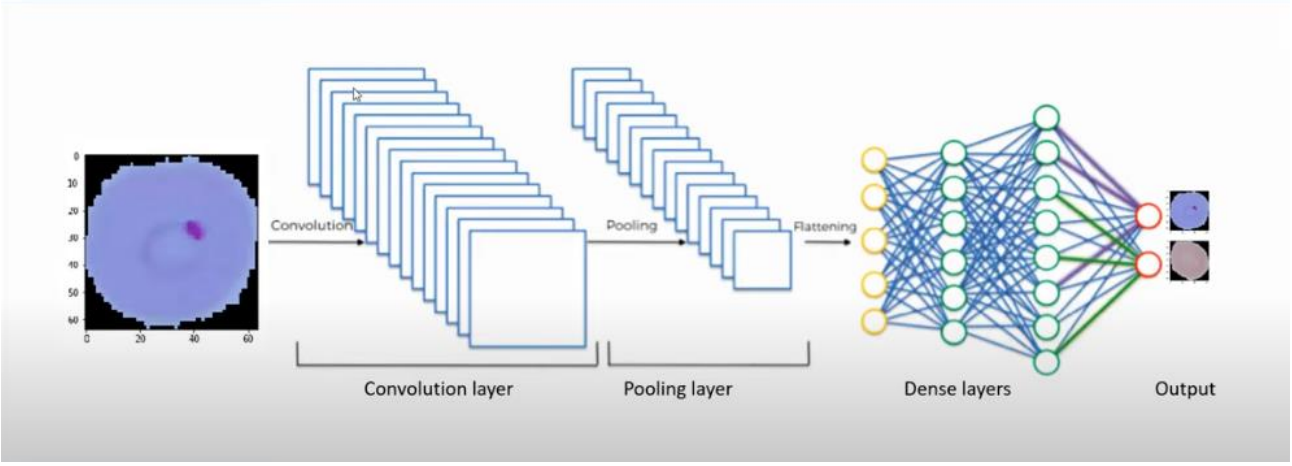
As the start point to reach the methodology am following there is no golden rule in deep learning u should experiment since my dataset isnt considered a huge one so for me I strated with 80% 20 % split.

Where the training split will be 80% for the training samples and 20% for the test samples

Model Architecture Selection

CNN

Simple CNN architecture used to classify



its a specialized type of neural networks and its a well suited for a tasks for image recognitions , classifications and regression tasks.

what makes CNN special when it comes into images automatically learns the features patterns from an input image through a multiple layers :

Convolutional layers

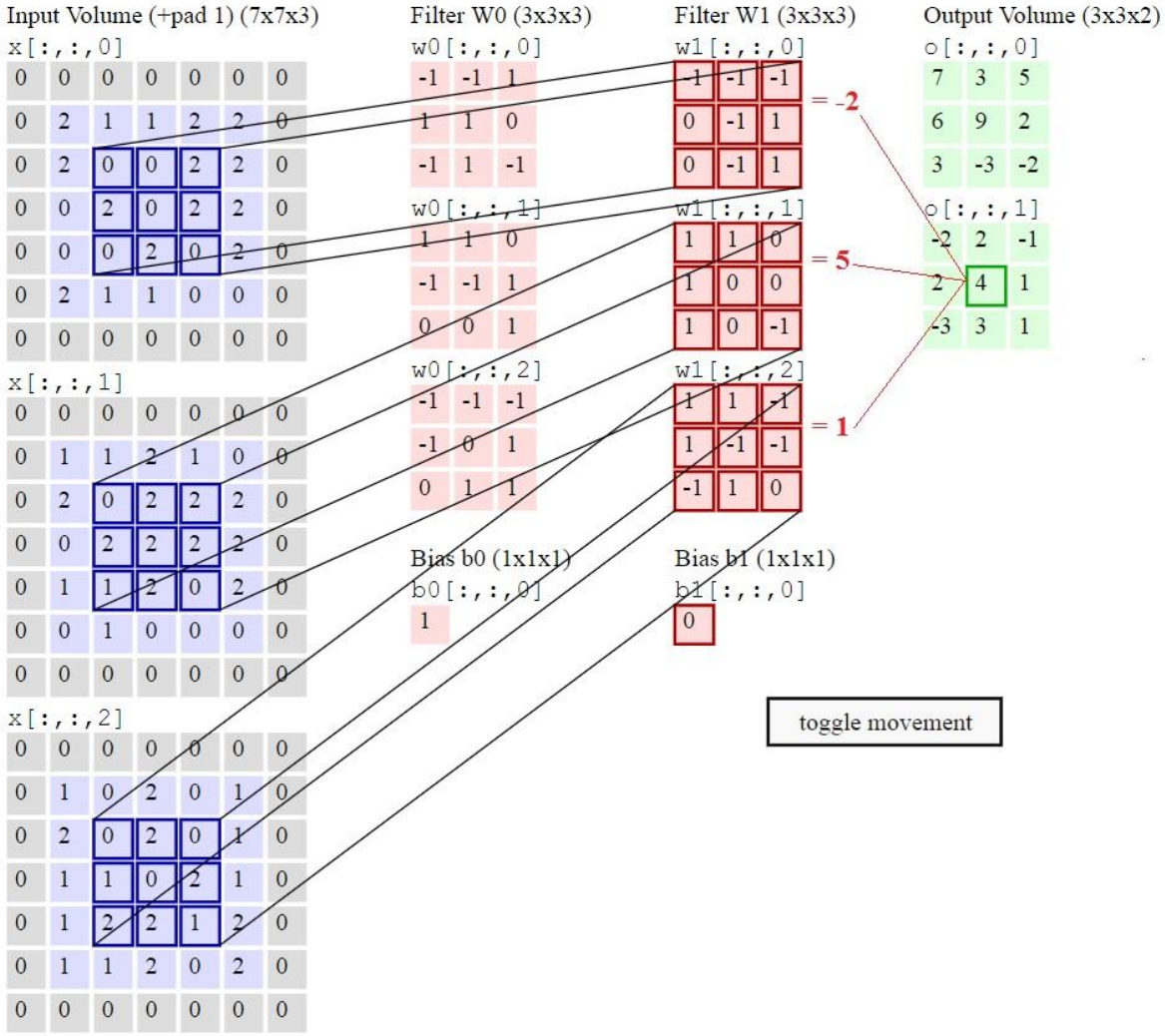
Pooling layers

Fully connected layer

Adding convolution and pooling layers helps improving the model accuarcy

What is Convolution ?

Is a process where a kernel or filter will be applied into an image in order to extract features



Convolution

The convolutional layers are one of the most important layers of CNNs, their basic function is to identify patterns or features in an image. CNNs link input images to the output by convolving them

with learnable filters.

can thus extract hierarchical representations that model the spatial pattern.

Pooling

Sometimes, through max pooling or average pooling, we need to reduce the size of the feature map, thereby reducing computational demand and need for translation in variance. These operations consolidate information and hold significant characteristics while decreasing spatial scales.

Padding

Padding is used in convolution to keep the shape of feature maps intact such that their sizes remain the same. Introduction Padding involves placing zeros around the input image or feature map so that the size of the output can match the size of the input so that features on the border can be learned.

Batch Normalization

Definition: Batch normalization is a technique of normalizing the inputs of a layer and it aims at normalizing and scaling activations. The process of normalizing the inputs applies to all layers and accelerates training, reduces internal covariate shift and is useful to improve generalization.

Training Stability: It helps to training stabilize by normalizing the activations according to distribution. It reduces the vanishing and exploding gradient issues, which makes it possible to more deeply and stably train the model.

Implementation: It is worth to note that CNN architectures often include batch normalization layers and these are added subsequent to convolutional layers and fully connected layers. These layers help in normalizing the activations within each mini-batch and aids in the quick convergence thus facilitating the speed up of training.

Dropout

It is a technique that is used in training a deep learning model where the neurons are randomly deactivated in the process known as dropout. Thus, dropout randomly drops some units and minimizes co-adaption among neurons while enhancing generalization.

It is integrated between fully connected layers, where neurons are randomly nullified concerning the dropout rate during training. Stochastic regularization is an effective technique in model generalization that relieves the problem of overfitting.

Research has proven that dropout increases model's generalization and performance across different DL tasks. Reducing the strength of signals in the training procedure, dropout contributes to the extraction of diverse features.

Flattening

Flattening is the conversion of feature maps which are arranged in a matrix format into vectors of size [batch_size × H × W × C] = 1. This layer is basically in-between the convolutional and fully connected layer and aid in changing one to the other especially when it is transitioning from a spatial hierarchy of features to a dense one.

When flattening, the spatial dimensions of the feature maps are reduced to one vector which contains some of the features and the spatial structure. The fully connected layers are featured in the next layer that takes this flattened representation as an input.

Dense Layers (Hidden Layers)

The fully connected or dense layers, learn abstract features and dependencies with the input data from a neural network. These layers press the input values in a non-linear fashion and hence through them difficult approximation functions can be easily modeled.

The density of layers is a decision where one has to decide about such aspects as, how many of these layers are to be 'hidden,' what number of neurons we are to have, and which type of activation functions ought to be used? Hence these parameters are useful in adapting of certain CNN architectures to certain tasks and the associated sets of data.

In the case of training, dense layers attempt at searching the dataset for some particular form of pattern and also tries to generalize it. Applying backpropagation and averaging for parameter tuning of dense layers so as to minimize the loss function of the model enhancing its performance.

Activation Functions

The non-linear transformations in the raw data are keyed on to by the activation functions on the

neurons of CNNs to enable the models learn complex patterns of the signals. There is a selection of basic activation functions inclusive of ReLU function, sigmoid function, and tanh functions whereby every function came with both strength point and weakness point.

Tasks are transformed by activation functions to non-linear signs from the necessity of, sums of inputs, weights, and biases so that CNNs able to fit and learn the non-linear and complex functions of data.

Being nonlinear in nature, the activation function perform on the neuron's weighted sum in relation to the inputs given. Consequently, by designating numerous layers into Convolutional Neural networks, they can make such predictions which cannot be made through linear models due to the presence of non-linear activation functions for depicting complex relational mechanisms.

Optimizer: Adam

ADAM is a popular optimization algorithm that has been used frequently in training CNNs; it is flexible due to its nature. They have an adaptive learning rate to avoid the network from taking longer time to converge and also incorporate momentum so that the gradients are noisy for better optimization.

Equal to SGD but uniquely mastering rates to slash convergence time and boost performance, HS versus HS classifies Adam as a first-order optimization algorithm. He was able to learn factor for the weight and another one for the activation and this propels the optimization process of the model a notch higher.

In the training of CNN, Adam alters all the parameters so as to control the first and second order of gradients if necessary and optimizes the learning rate at the same time during the training of the model. Furthermore, the use of momentum as well as the modified learning rate ensures that the algorithm converges faster and also optimizes for a broader range of models.

Loss Function and Metrics

In regression tasks, loss functions aim at measuring the discrepancy between the predicted value of the target data and its actual value. MAE stands for Mean Absolute Error and is one of the simplest and most often used loss functions in age regression tasks, where the criterion it measures is the average absolute difference between the predicted and the real value of age.

Measures such as MAE convey the start and endpoint of the CNN models, as well as the dependability and precision of its predictions. The higher value of N indicates the greater number of years where the model had been able to estimate the age of the patient closer to the actual age with minimal variation which is determined by the lower of the two specific MAE values.

Specifically, the set of loss functions to be used in learning and the criteria for comparing performance of the resulting models depend on the task characteristics and properties of a given set of data as well as on the particular model to be applied. By thus specifying which aspects of the CNN should be evaluated, they have also made clear how the CNN models should be evaluated and what directions should be taken to build and develop the model.

Hyperparameters

hyperparameters refer to the CNN model parameters that are a bit more general in the sense that they control the processes and outcomes of the models. . Other fundamental hyperparameters consist of the number of epochs, the number of samples to pass through at once, the learning rate as well as the structure of the neural networks used to train the models, which dictate the training patterns and convergence of the models.

Epochs is the number of passes, and fine epoch is the number of training cycles done over the entire train set during the training session. The more epoch means that the machine will learn more number of patterns of the data we are feeding and such kind of epoch has an added advantage to CNN models at most of the times; this type of epoch takes much time in training compare with others.

Batch size the number of samples that are passed through or operated on by the training algorithm simultaneously. The large batches allow for the use of parallel operations and therefore faster training of a given model while the small batches enhance the stability of the training especially in their structural form.

Understanding Overfitting and Underfitting in Convolutional Neural Networks (CNNs)

Convolutional neural networks (CNNs) are deep learning models that are capable of extracting high-level features and matching complex patterns in the image data population. However, during the training process, CNNs may encounter two common problems:

overfitting — a situation where a model fits all the data too well, creating more noise than signal and making predictions less accurate, and; underfitting — this is a scenario whereby the data is poorly fitted by the model due to the exclusion of an important feature. These are due to the model complexity, the characteristics of the data, and the procedures of the training.

Overfitting

But if the CNN model tries to learn the characteristics of the training data in detail, it starts overfitting and instead of identifying the relation between the input and output, it starts identifying noise present in the training data. It therefore does very well for all the training examples and poorly for all other, unseen examples if any.

There are number of issues in CNNs that facilitate overfitting, including:

Excessive model complexity, CNNs with many parameters; they can prone to capture noise and other spurious correlations from the training data.

A small dataset may not seem diverse and varied enough to properly train a set of models that will not over-fit the data set.

These basic strategies, including dropout and weight decay, do not sufficiently curb the model .

Some of the bad signs of overfitting include:

Poor generalization: When training a model, several times it becomes excessively tailored to understanding the training data, meaning that it will not do well with new data it comes across.

Underfitting

Is a state that occurs when a CNN model is unable to learn from the training data or fail to capture the then relevant patterns or relationship in the data that are supposed to be used for predicting future patterns. These the indications may be; inadequate model complexity or time spent in the process of training the model hence straining the training capability of the model.

Some of the things that cause underfitting include the following:

Insufficient model complexity: The CNNs with less capacities cannot analyze the data that is being fed into it and that is why during the training and validation they prove to be futile.

Inadequate training duration: It implies that the model cannot map the function that is optimum for the data since the learning process is constrained by epochs or the maximum amount of data used in one step making the representation to be underfitted.

Inability to achieve a model that is complex enough to capture all the relations in the dataset leads to underfitting The mentioned problems include:

Poor performance: This is evidenced by low accuracy and the high error rate when the models and networks are tested on the training and validation datasets The high error rate when testing an underfitting model on the various sets of data is due to the fact that it does not recognize the key factors that govern a certain dataset.

Limited expressiveness: If the model is underfitted, it does not have enough capacity to learn about the complexity and shape of the true distribution of the data and hence its prediction capability is not excellent.

Inability to learn: This is because as presented in the model above, the DNN does not have the ability to derive proper features of the data that are essential in allowing it to make forecasts and make proper sense of the data.

Transfer learning

1. Introduction

Transfer learning is one of the overlaid important concepts in Machine Learning and Deep Learning particularly in Convolutional Neural Networks (CNNs) which utilises the knowledge acquired from previous models to solve other related problems more efficiently. Transfer learning involves taking a model trained on the source domain and typically fine-tunes it to work on the target domain with significantly less labeled data.

2. Basic Concept

Transfer learning is based upon the concept that knowledge that is learned on one occasion is helpful on another occasion. However, instead of training CNN model C on data set D from scratch, which need a large of labeled data and calculating resource, transfer learning tries to reuse features learned by pre-trained CNN model C on large-scale data set such as ImageNet. This means that, in addition to using the presented algorithm to perform the primary classification task, the model has learned features that can be utilized to successfully perform other, related tasks with little labeled data.

3. Workflow

The typical workflow of transfer learning involves the following steps:The typical workflow of transfer learning involves the following steps:

Pre-trained Model Selection: Select a fine-tuned CNN model for the target task as per the characteristics including architecture, performance and domain relevance.

Feature Extraction: They pointed out that one should freeze the convolutional layers of the pre-trained model to extract features of the input images. These are used as features of high level, which defines meaningful patterns and structures.

Fine-tuning: In additional, it is often helpful to train the higher level layers of the pre-trained model on the target task's data. We can fine-tune the model to perform better regarding the specifics of the target domain by applying fine-tuning.

Training and Evaluation: In this case, number of iterations during training the adapted model or model modification is generally lower and learning rate is decreased, if it is trained on the dataset of the target task. After constructing the model, it should be assessed on the validation set and the hyper-parameters then adjusted where necessary.

4. Benefits

Transfer learning offers several advantages, including:Transfer learning offers several advantages, including:

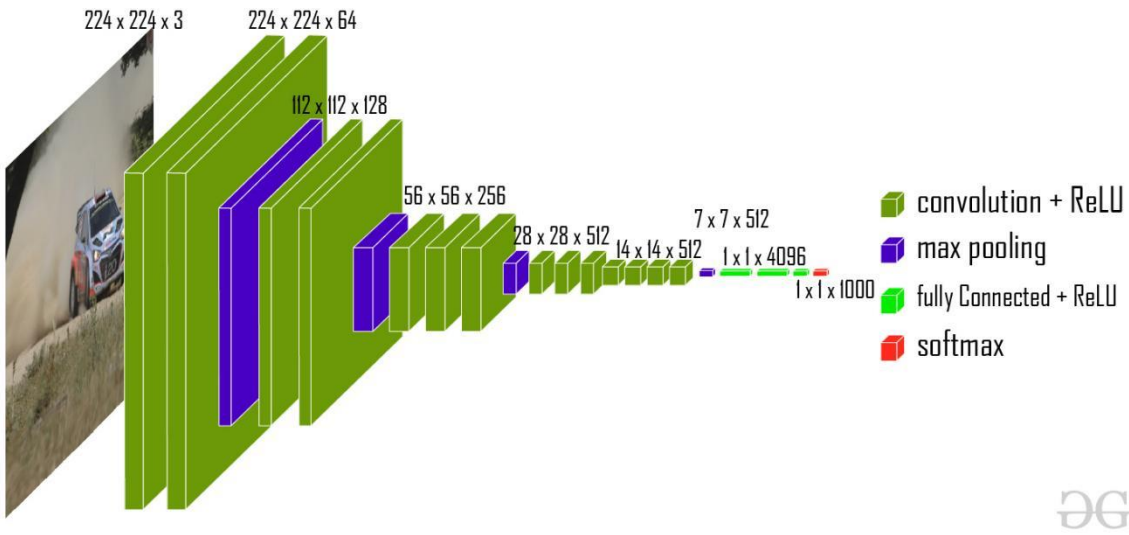
Reduced Training Time: Because of its ability to reuse models that have already been trained, transfer learning saves degrees of freedom, computational resources, and time particularly when working with tasks that have a limited amount of labeled data.

Improved Generalization: Pre-trained models seem to learn basic forms from the generic forms within large-scale forms, which helps when solving specific forms for new tasks, even if there is limited data available.

Pre-Trained models

Some of the pre-trained model used in my Project explaining the architecture

Ex: VGG16



Here's a breakdown of the **VGG-16** architecture based on the provided details:

Input Layer:

Input dimensions: (224, 224, 3)

Convolutional Layers (64 filters, 3×3 filters, same padding):

Two consecutive convolutional layers with 64 filters each and a filter size of 3×3.

Same padding is applied to maintain spatial dimensions.

Max Pooling Layer (2×2, stride 2):

Max-pooling layer with a pool size of 2×2 and a stride of 2.

Convolutional Layers (128 filters, 3×3 filters, same padding):

Two consecutive convolutional layers with 128 filters each and a filter size of 3×3.

Max Pooling Layer (2×2, stride 2):

Max-pooling layer with a pool size of 2×2 and a stride of 2.

Convolutional Layers (256 filters, 3×3 filters, same padding):

Two consecutive convolutional layers with 256 filters each and a filter size of 3×3.

Convolutional Layers (512 filters, 3×3 filters, same padding):

Two sets of three consecutive convolutional layers with 512 filters each and a filter size of 3×3.

Max Pooling Layer (2×2, stride 2):

Max-pooling layer with a pool size of 2×2 and a stride of 2.

Stack of Convolutional Layers and Max Pooling:

Two additional convolutional layers after the previous stack.

Filter size: 3×3.

Flattening:

Flatten the output feature map (7x7x512) into a vector of size 25088.

Fully Connected Layers:

Three fully connected layers with ReLU activation.

First layer with input size 25088 and output size 4096.

Second layer with input size 4096 and output size 4096.

Third layer with input size 4096 and output size 1000, corresponding to the 1000 classes in the ILSVRC challenge.

Softmax activation is applied to the output of the third fully connected layer for classification.

This architecture follows the specifications provided, including the use of ReLU activation function and the final fully connected layer outputting probabilities for 1000 classes using softmax activation.

Process and results :

1- First of all I started with some starting point as a beginning point for testing to tablute the result I went first of all into a customed CNN and seeing the different result and comparing with when am adding the pre-trained models into work since my dataset is kind of not huge so seeing the differences of the MAE result

First tuned the hyperparameters at a certain point as startup where I will apply the methodology am following I runned a couple of training usig different splits , different batch sizes , different amount of dense layer , trained a customed CNN from scratch to see how it behaves with UTKFace dataset and aiming to decrease the MAE of regression value by applying the Transfer learning into the work where I saw the good differences and performance and took the lowset MAE score and hyper tuned the hyperparameters seeing the good improvement and the overfit reduction by experimenting different models.

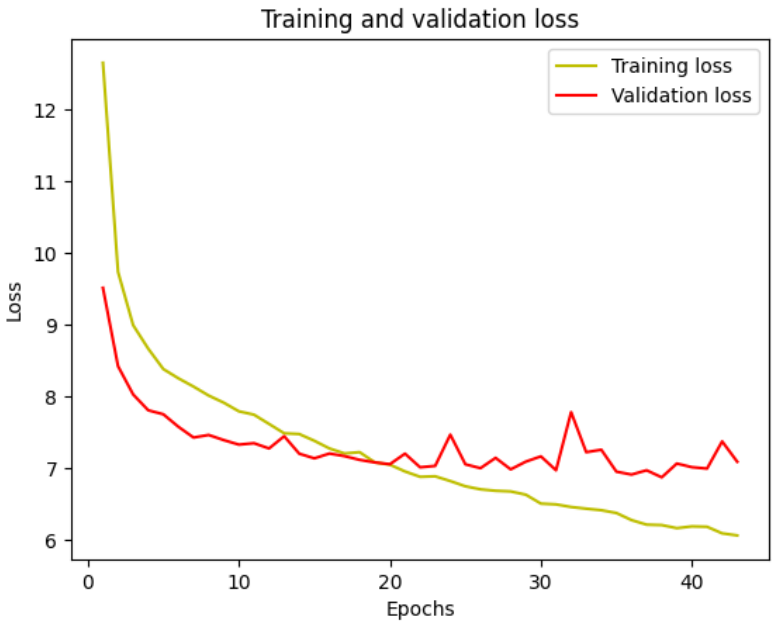
Version 1 of the first Trainings starting with a 5 dense layer and batch normaization layers in between and adding some of the regularization technique like Drop out split 80% 20% took and 85% 15% into the experiment

VGG16 with a score of :

112/112 [==============================] - 6s 51ms/step - loss: 8.2640 - mae: 8.2640

Test Loss: 8.26401424407959

Test MAE: 8.26401424407959

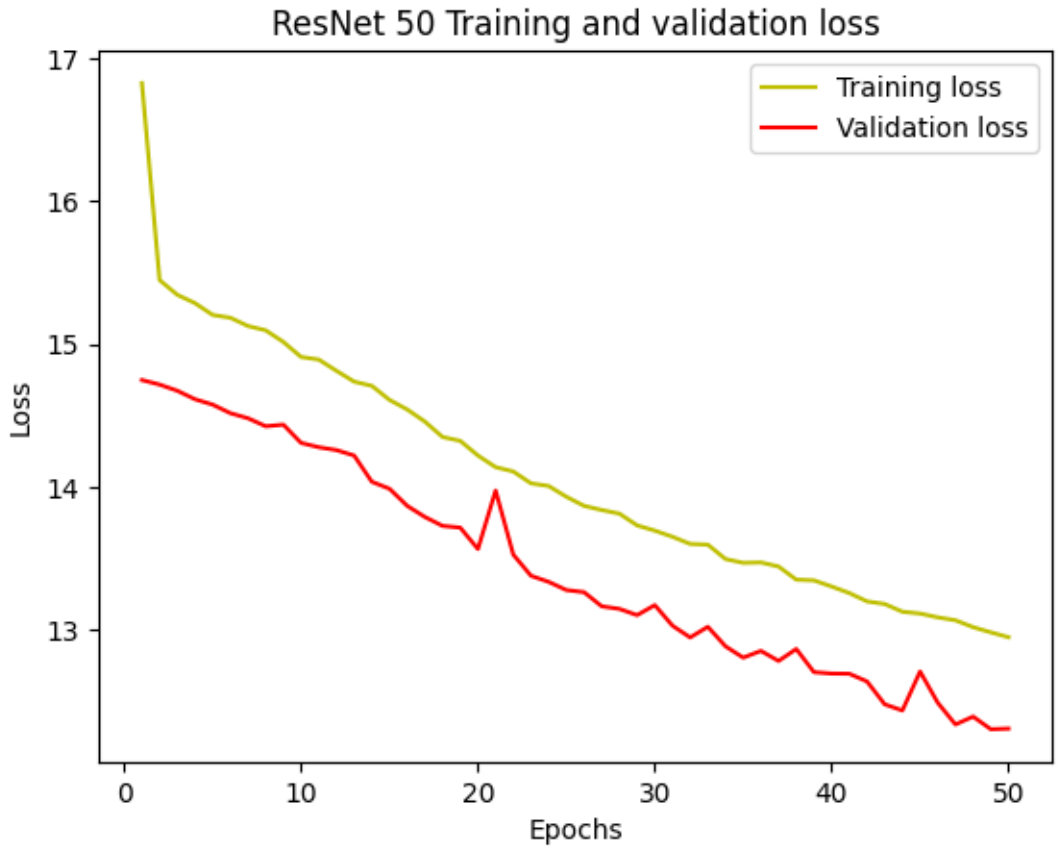


ResNet50 with a score of :

112/112 [==============================] - 4s 35ms/step - loss: 12.3066 - mae: 12.3066

Test Loss: 12.306586265563965

Test MAE: 12.306586265563965

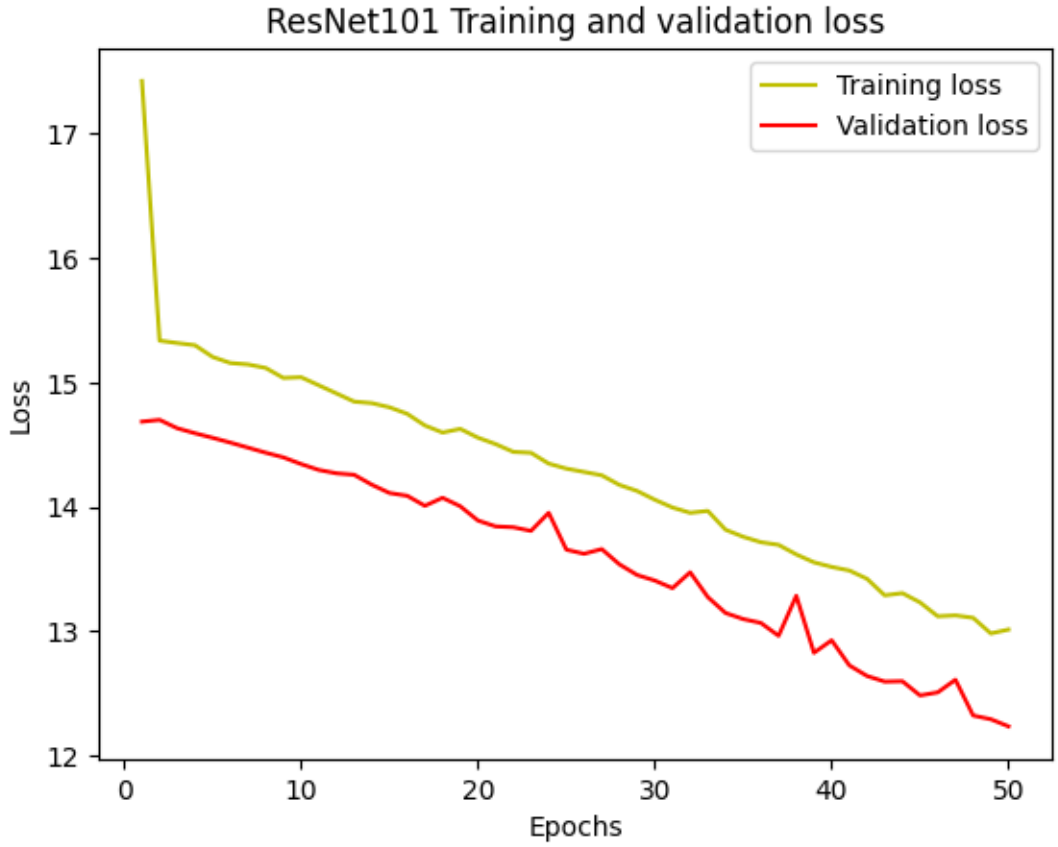


ResNet101 with a score of :

112/112 [==============================] - 6s 57ms/step - loss: 12.2355 - mae: 12.2355

Test Loss: 12.235481262207031

Test MAE: 12.235481262207031

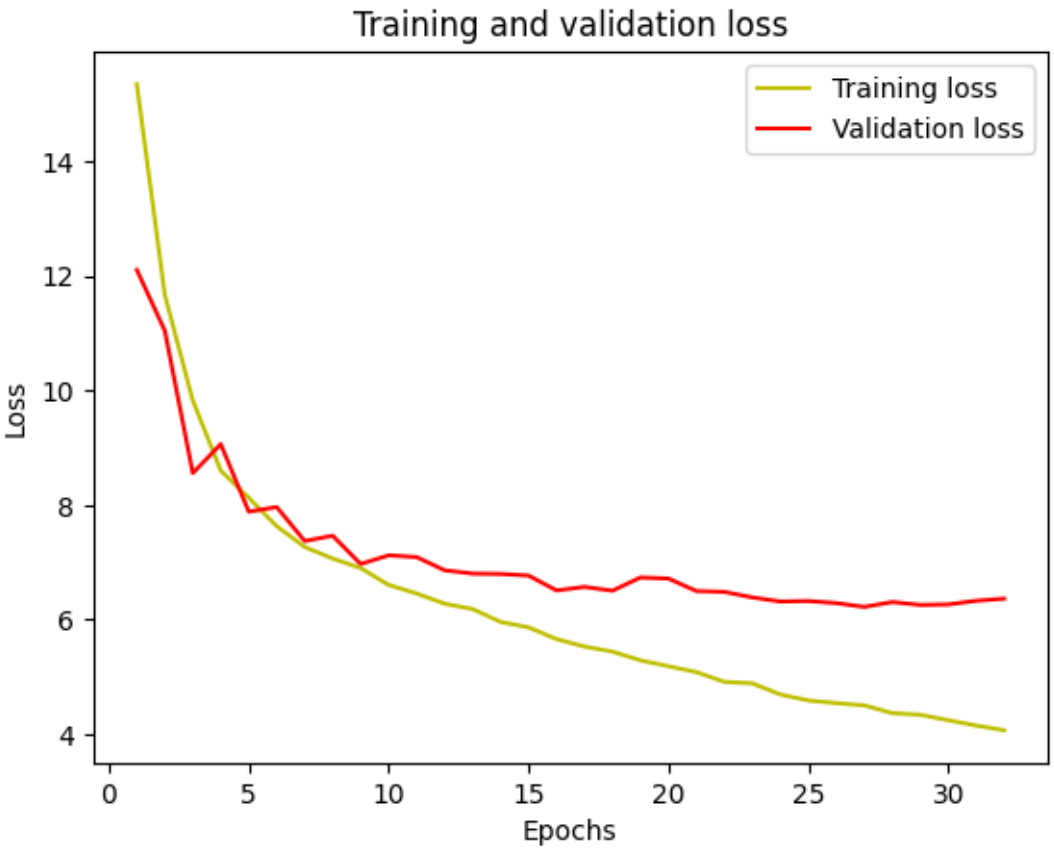


VGG19 with a score of :

186/186 [==============================] - 10s 54ms/step - loss: 8.3671 - mae: 8.3671

Test Loss: 8.367077827453613

Test MAE: 8.367077827453613



Using pre-trained with a kind of splits and adding dense layers , batchnormalization , dropout layers and different rates after training a lot of trainings for UTKFace

Table of verion 1

Pre-trained	**VGG16**	**VGG19**	**ResNet50**	**ResNet101**
BatchNormalization layers	3	3	3	3
Dense layers	5	5	5	5
Dropout layers(rate)	2(0.3)	2(0.3)	2(0.3)	2(0.3)
Test split	15%	15%	15%	15%
MAE score	**8.26**	8.36	12.30	12.23

Results of published paper using DCNN's [1]

Table 2 Classification and regression results of UTKFace

Method	Classification accuracy			MAE of regression
	Individual	5 years grouping	10 years grouping	
ResNet18	0.89	0.95	0.98	**9.19**
ResNet34	0.68	0.89	0.96	9.65
ResNet50	**0.89**	**0.97**	**0.99**	9.66
Inceptionv3	0.43	0.82	0.93	9.50
DenseNet	0.76	0.94	0.99	**9.19**

2- The second batch run I started with some starting point as a beginning point for testing to tablute the result I went first of all into a customed CNN and seeing the different result and comparing with when am adding the pre-trained models into work since my dataset is kind of not huge so seeing the differences of the MAE results.

After understanding the differneces and how the improvement techniques worked with the UTKFace and the overfit prevention and managing the MAE score to get lower score. First I tuned the hyperparameters at a certain point as startup where I will apply the methodology am following I runned a couple of training usig different splits , different batch sizes , different amount of dense layer , trained a customed CNN from scratch to see how it behaves with UTKFace dataset and aiming to decrease the MAE of regression.

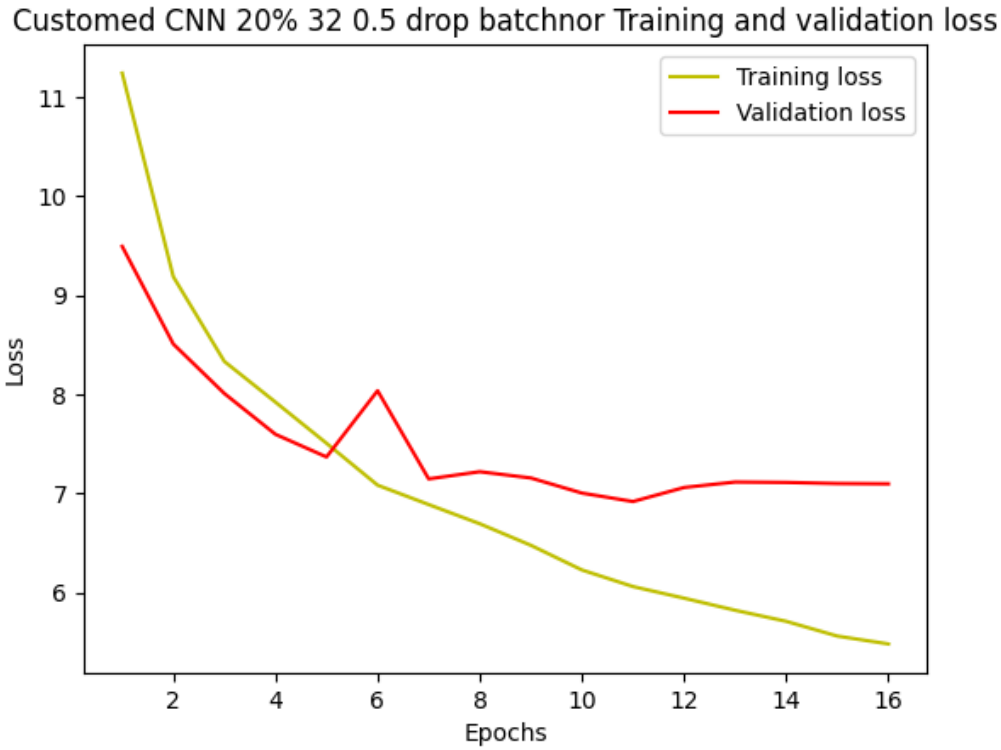
started with 20 epochs and 1024 dense layer neuron first layer in ANN 20% test after this step added a dropout layer of a rate 0.5(50%) then BatchNormalization layers with a early stopping function after 5 epochs patience when after 5 epochs the validation loss didn't improve.

Customed CNN

149/149 [==============================] - 1s 10ms/step - loss: 9.5860 - mae: 9.5860

Test Loss: 9.2013317108154

Test MAE: 9.2013317108154

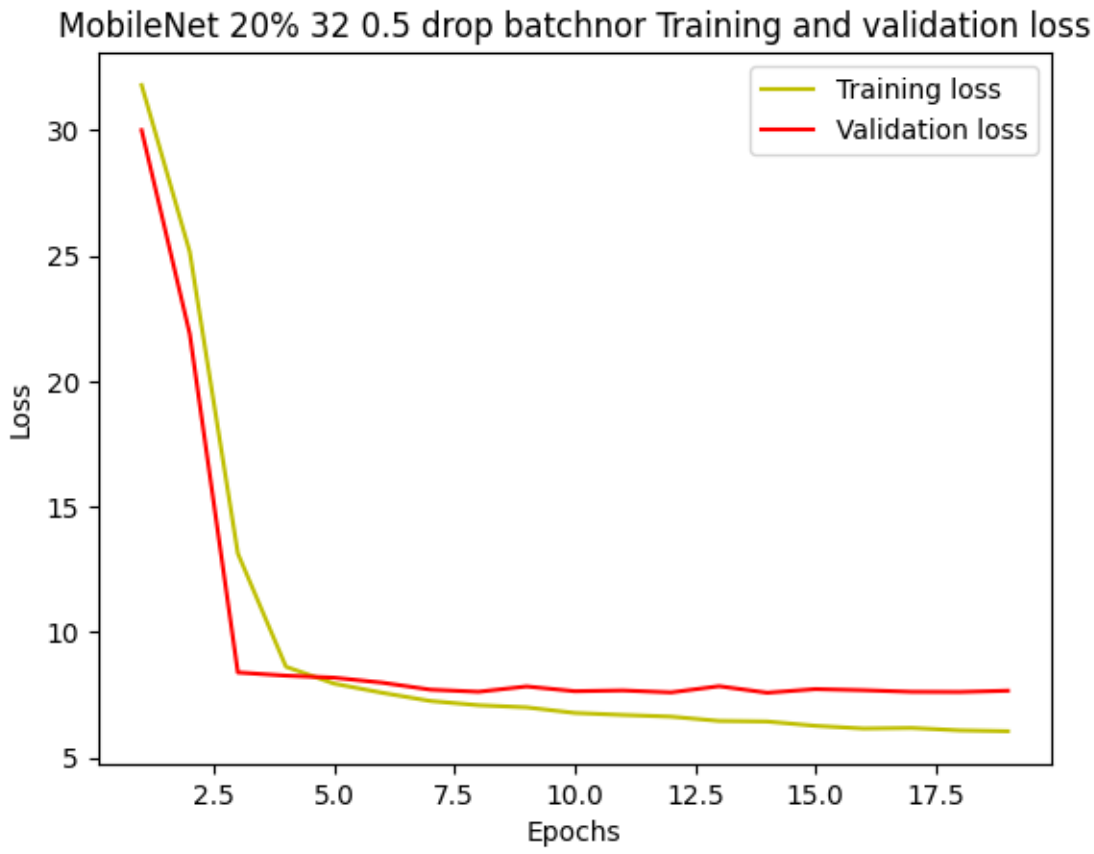


MobileNet

149/149 [==============================] - 1s 10ms/step - loss: 7.5860 - mae: 7.5860

Test Loss: 7.586013317108154

Test MAE: 7.586013317108154

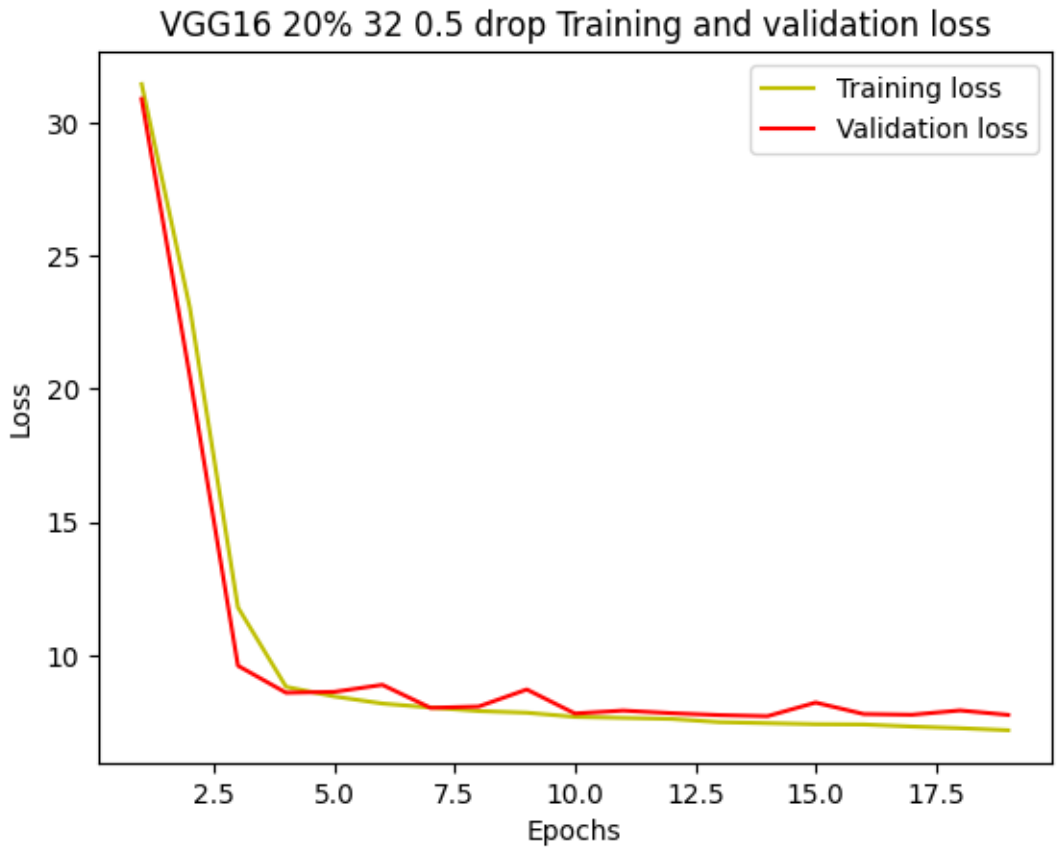


VGG16

149/149 [==============================] - 2s 12ms/step - loss: 7.7170 - mae: 7.7170

Test Loss: 7.716973781585693

Test MAE: 7.716973781585693

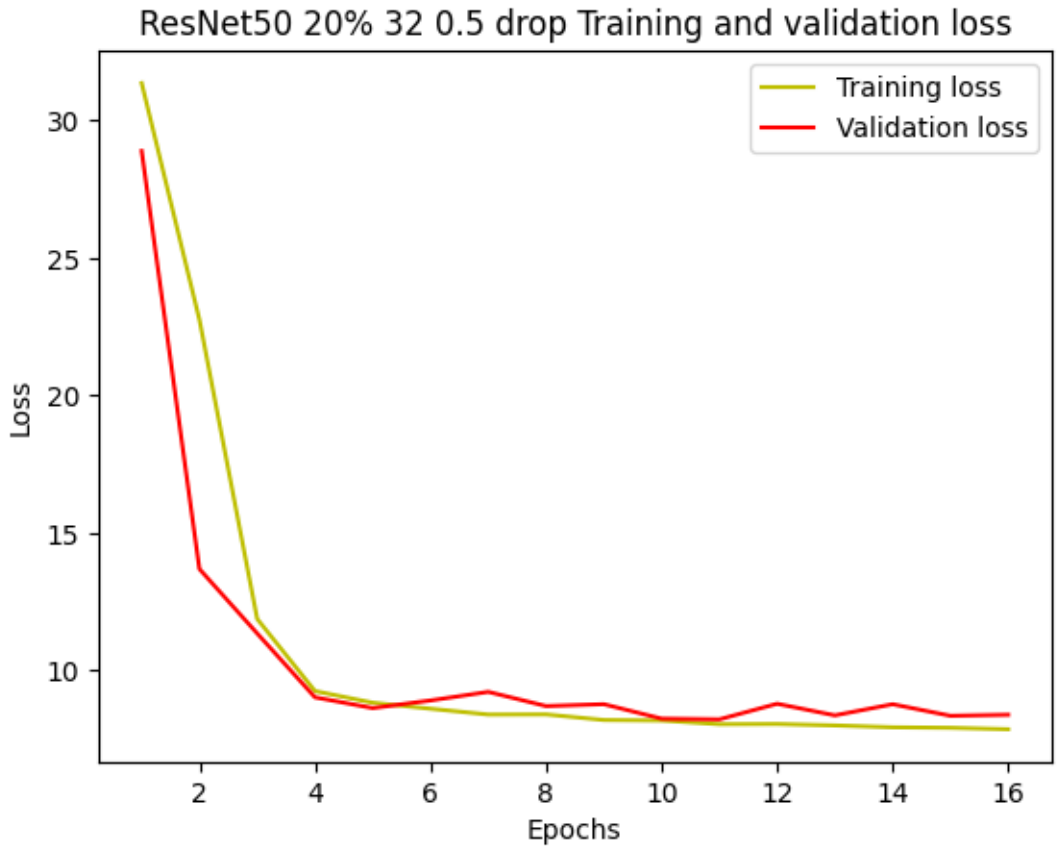


VGG19

149/149 [==============================] - 2s 13ms/step - loss: 8.2222 - mae: 8.2222

Test Loss: 8.222183227539062

Test MAE: 8.222183227539062



DenseNet121

149/149 [==============================] - 3s 18ms/step - loss: 7.0545 - mae: 7.0545

Test Loss: 7.054525375366211

Test MAE: 7.054525375366211

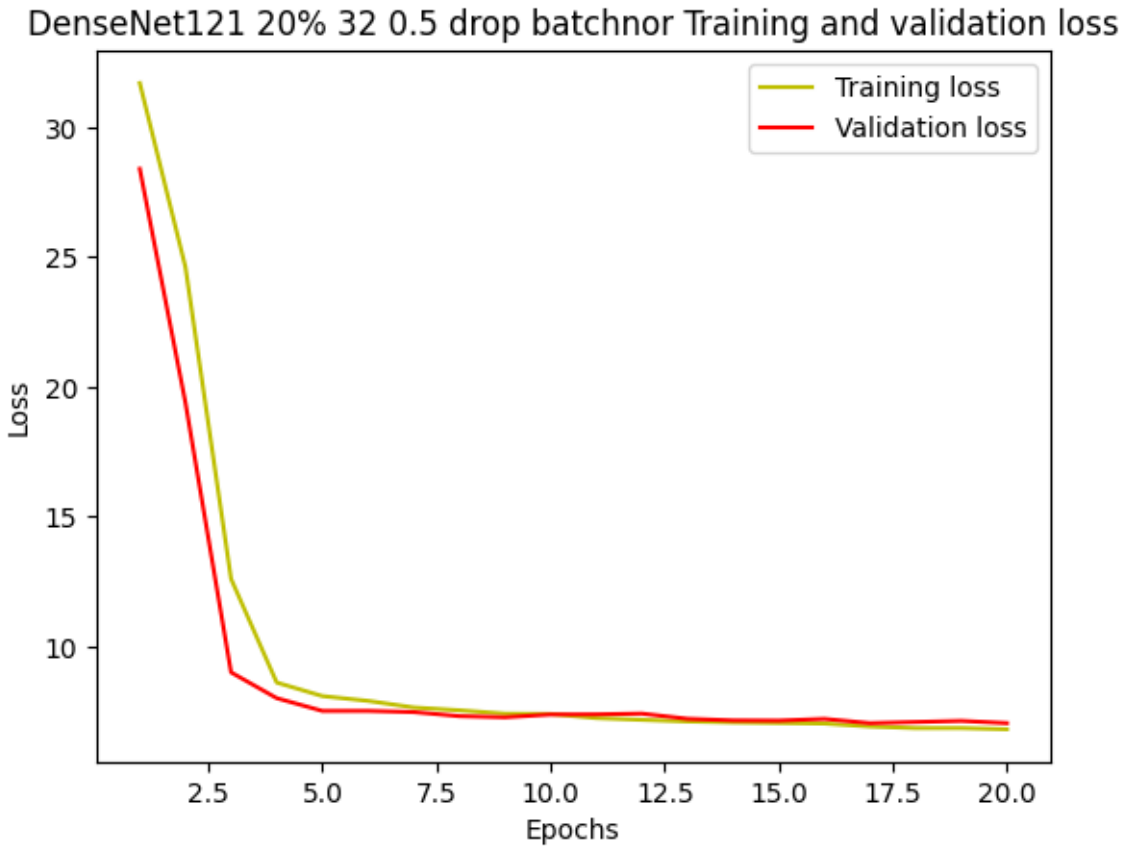


Table of version 2

Pre-trained	**VGG16**	**VGG19**	**MobilNet**	**DenseNet121**	**Customed CNN**
BatchNormalization layers	1	1	1	**1**	1
Dense layers	2	2	2	**2**	2
Dropout layers(rate)	1(0.5)	1(0.5)	1(0.5)	**1(0.5)**	1(0.5)
Test split	20%	20%	20%	**20%**	20%
MAE score	**7.77**	8.22	7.58	**7.05**	9.20
Batch size	32	32	32	**32**	32

Results of published paper using DCNN's [1]

Table 2 Classification and regression results of UTKFace

Method	Classification accuracy			MAE of regression
	Individual	5 years grouping	10 years grouping	
ResNet18	0.89	0.95	0.98	**9.19**
ResNet34	0.68	0.89	0.96	9.65
ResNet50	**0.89**	**0.97**	**0.99**	9.66
Inceptionv3	0.43	0.82	0.93	9.50
DenseNet	0.76	0.94	0.99	**9.19**

The published paper used the split of 90% training and 10% testing as we can see using DCNN fine tuning the pre-trained model. So I considered using my proposed method getting a lower MAE scores the method they followed for UTKFace benchmark dataset.

After getting the best behaved and best MAE results compared to other pre-trained models we got the **DenseNet121** taking it to tuning hyperparameters and seeing the difference and trying to reduce the overfits and MAE score to its minimum.

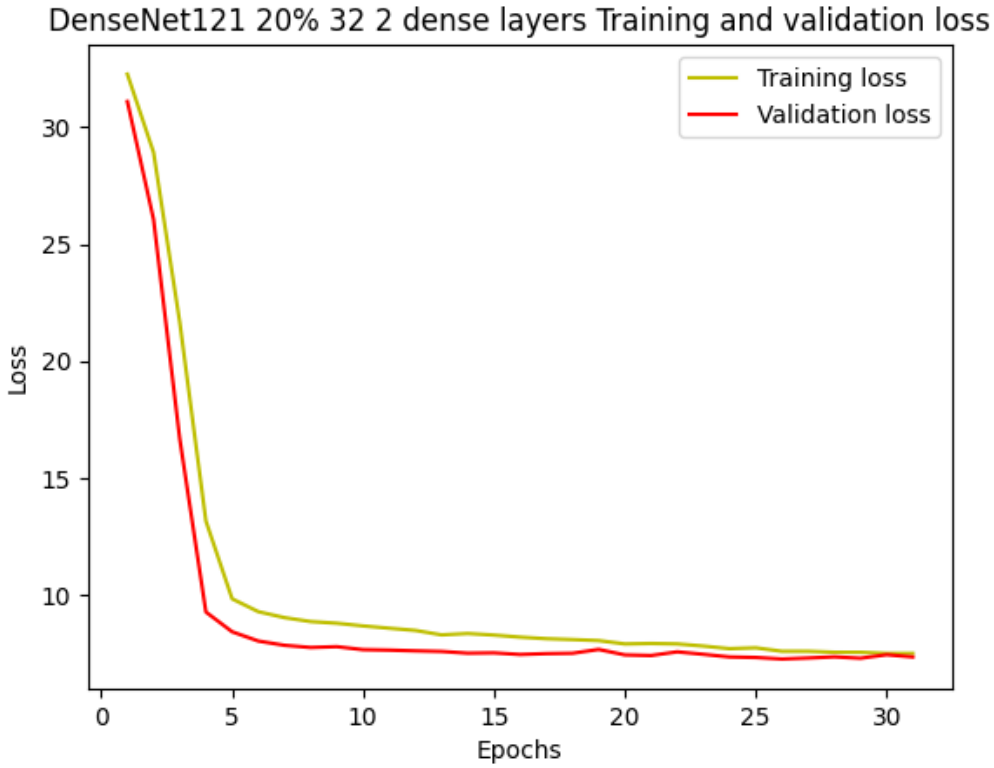
Changing the training and test split seeing how the UTKFace benchmark will react and reduces the MAE result and applying different dropoutrates and batchnormalization and batch sizes.

DenseNet121 (20% test)- 2 dense layers , dropout rate 0.4 , shuffling , random state 42 , batch normalization , batch 32 .

149/149 [==============================] - 3s 18ms/step - loss: 7.8845 - mae: 7.8845

Test Loss: 7.884525375366211

Test MAE: 7.884525375366211

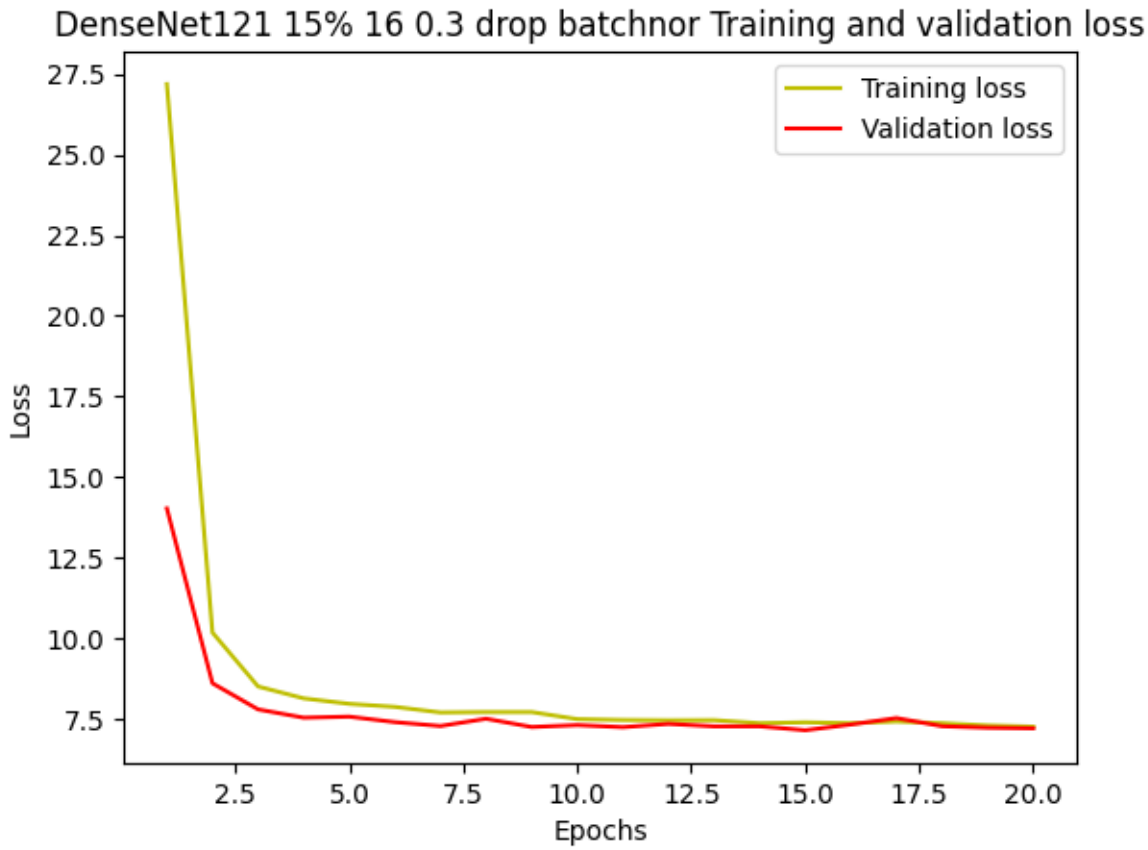


DenseNet121 (15% test)- 2 dense layers , dropout rate 0.3 , shuffling , random state 42 , batch normalization , batch 16 .

112/112 [==============================] - 3s 24ms/step - loss: 7.1484 - mae: 7.1484

Test Loss: 7.148413181304932

Test MAE: 7.148413181304932

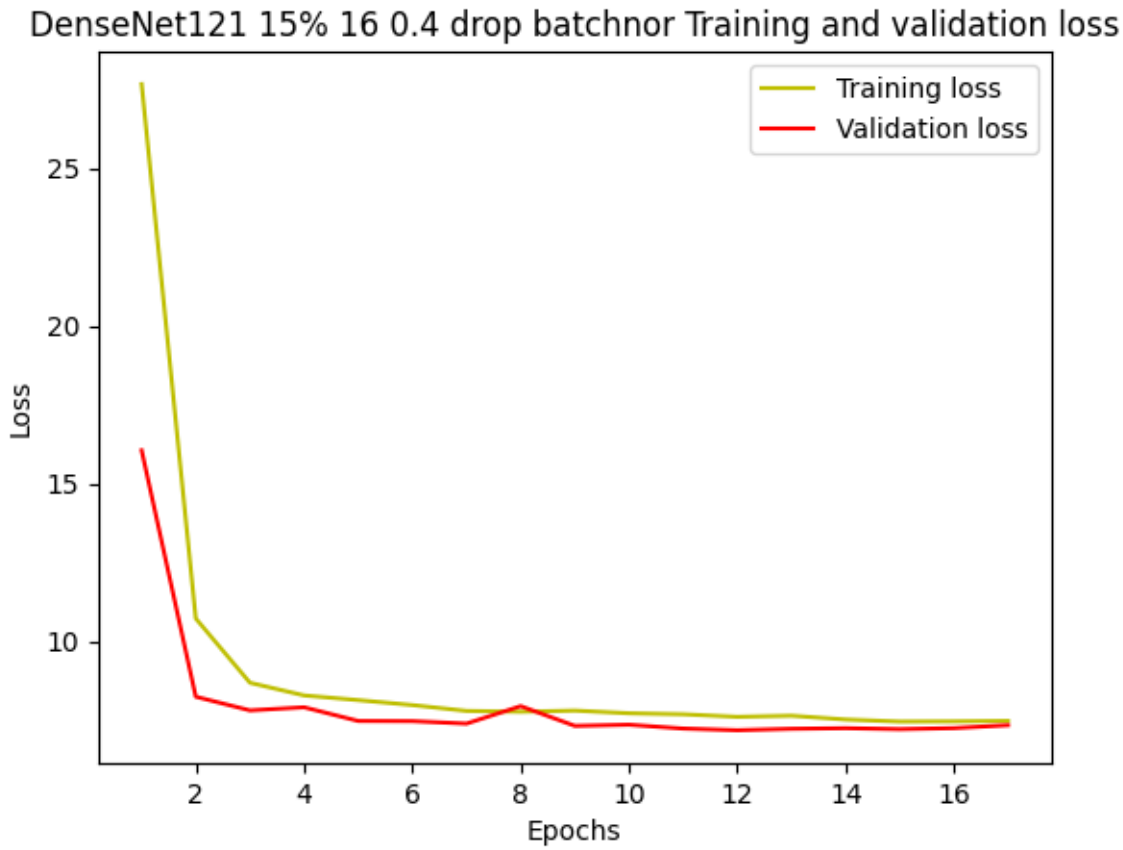


DenseNet121 (15% test)- 2 dense layers , dropout rate 0.4 , shuffling , random state 42 , batch normalization , batch 16 .

112/112 [==============================] - 3s 25ms/step - loss: 7.1703 - mae: 7.1703

Test Loss: 7.17030668258667

Test MAE: 7.17030668258667

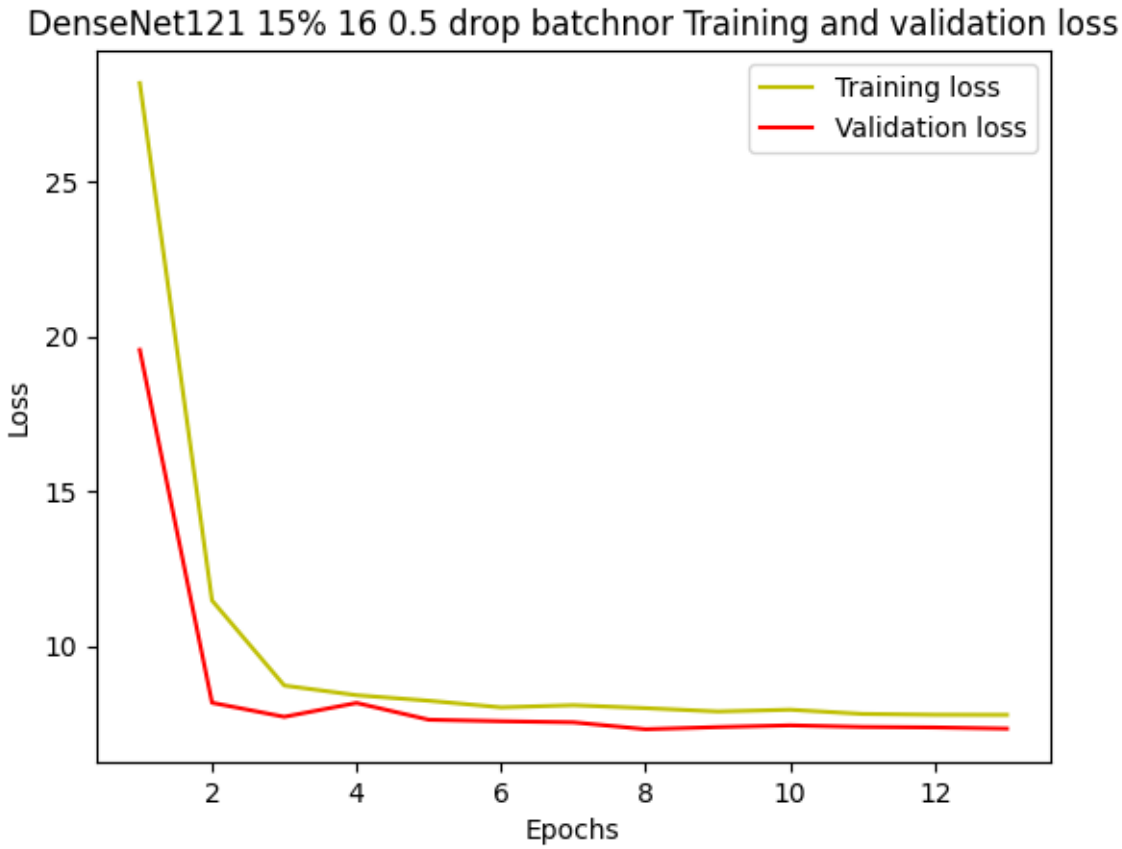


DenseNet121 (15% test)- 2 dense layers , dropout rate 0.5 , shuffling , random state 42 , batch normalization , batch 16 .

112/112 [==============================] - 3s 23ms/step - loss: 7.3225 - mae: 7.3225

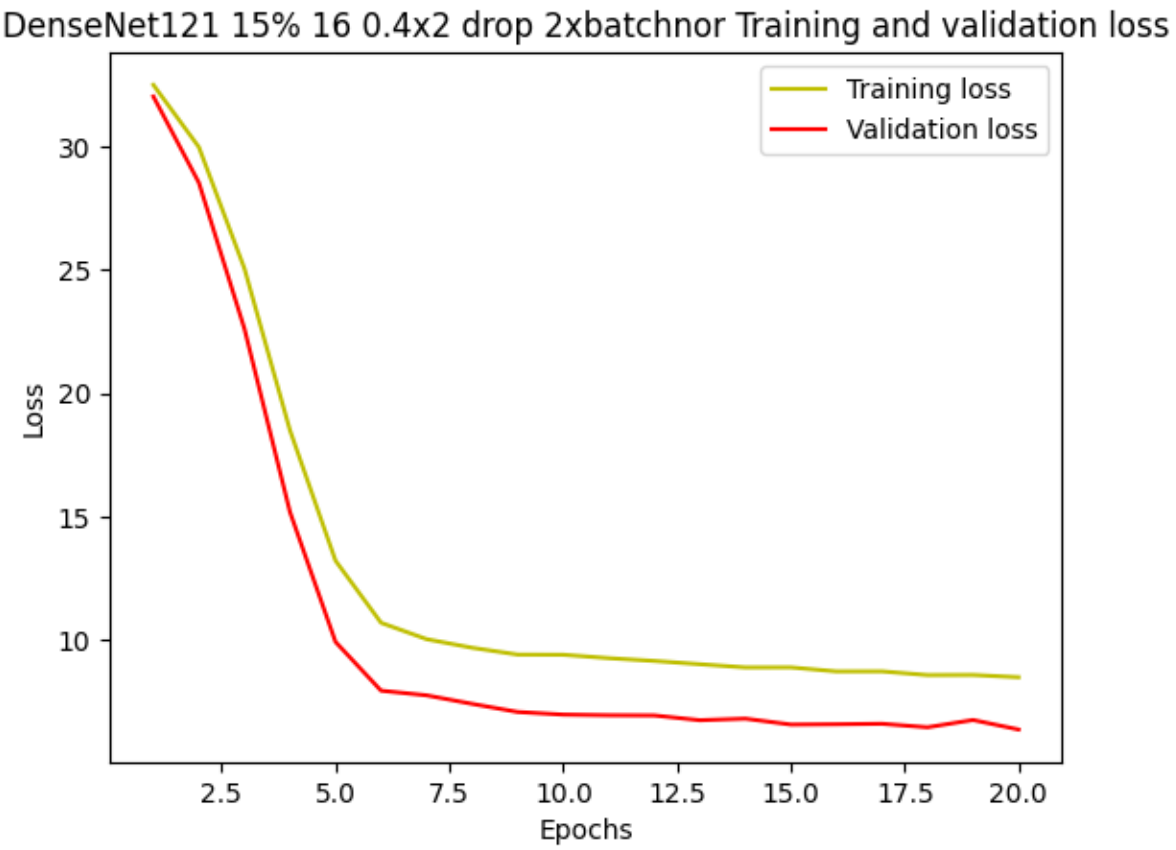
Test Loss: 7.322451591491699

Test MAE: 7.322451591491699


DenseNet121 15% 16 0.5 drop batchnor Training and validation loss
Training loss
Validation loss
Loss
25
20
15
10
2
4
6
8
10
12
Epochs

DenseNet121 (15% test)- 5 dense layers , 2 dropout rate 0.4 , shuffling , random state 42 , 2 batch normalization layers , batch 16 .

Started underfitting.

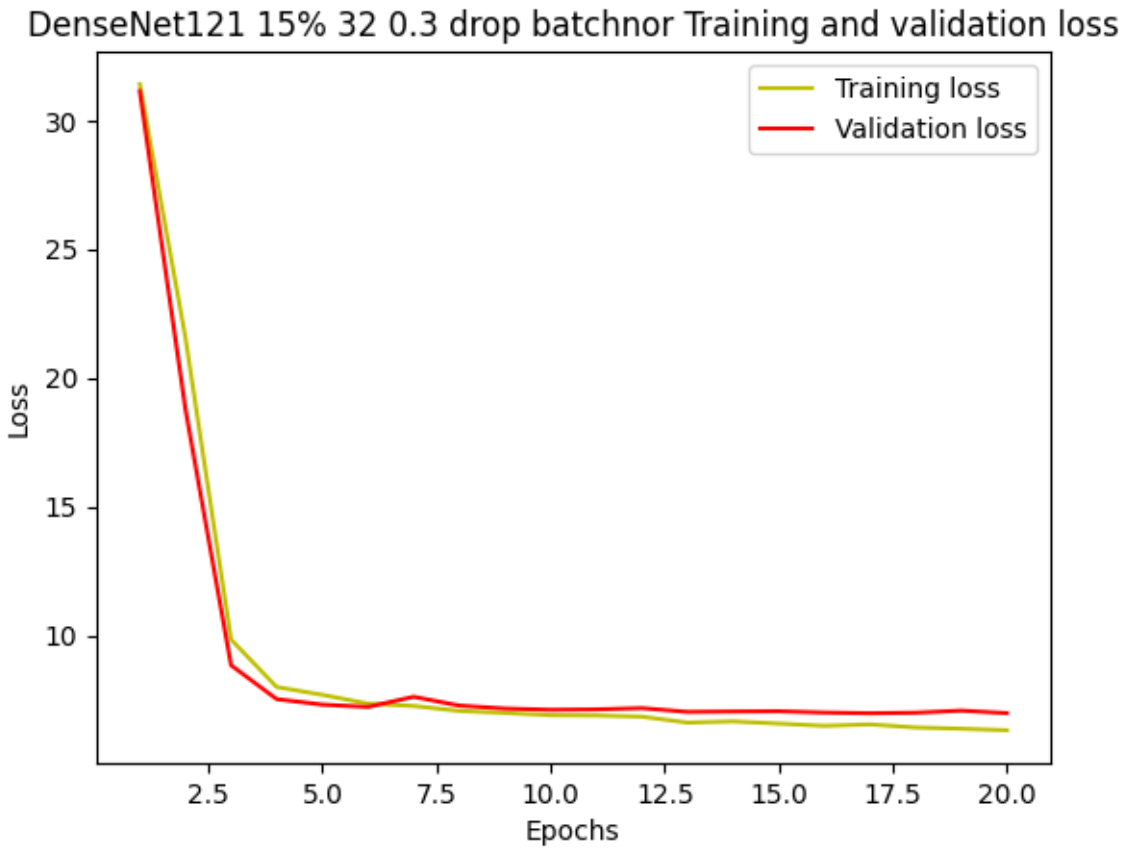


DenseNet121 (15% test)- 3 dense layers , 1 dropout rate 0.3 , shuffling , random state 42 , 1 batch normalization layer , batch 32 .

112/112 [==============================] - 3s 25ms/step - loss: 7.0678 - mae: 7.0678

Test Loss: 7.067783355712891

Test MAE: 7.067783355712891

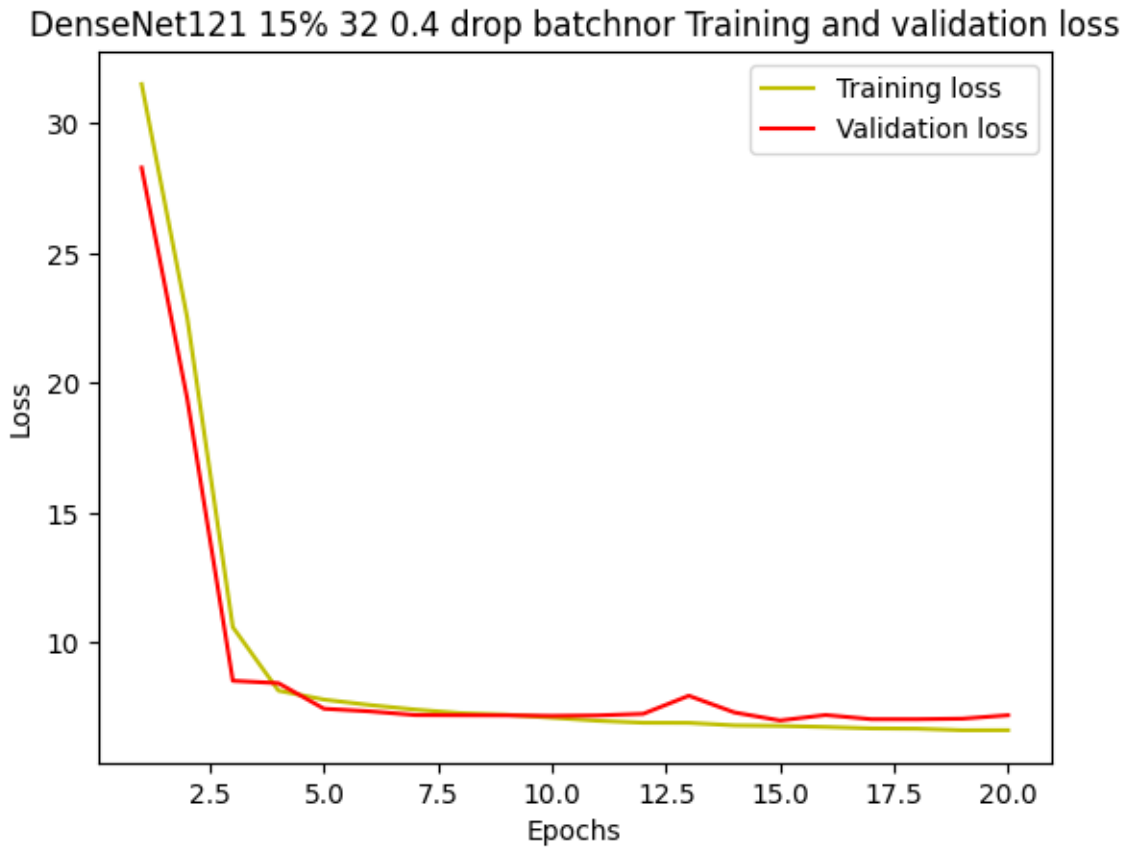


DenseNet121 (15% test)- 3 dense layers , 1 dropout rate 0.4 , shuffling , random state 42 , 1 batch normalization layer , batch 32 .

112/112 [==============================] - 3s 25ms/step - loss: 6.9997 - mae: 6.9997

Test Loss: 6.999687194824219

Test MAE: 6.999687194824219



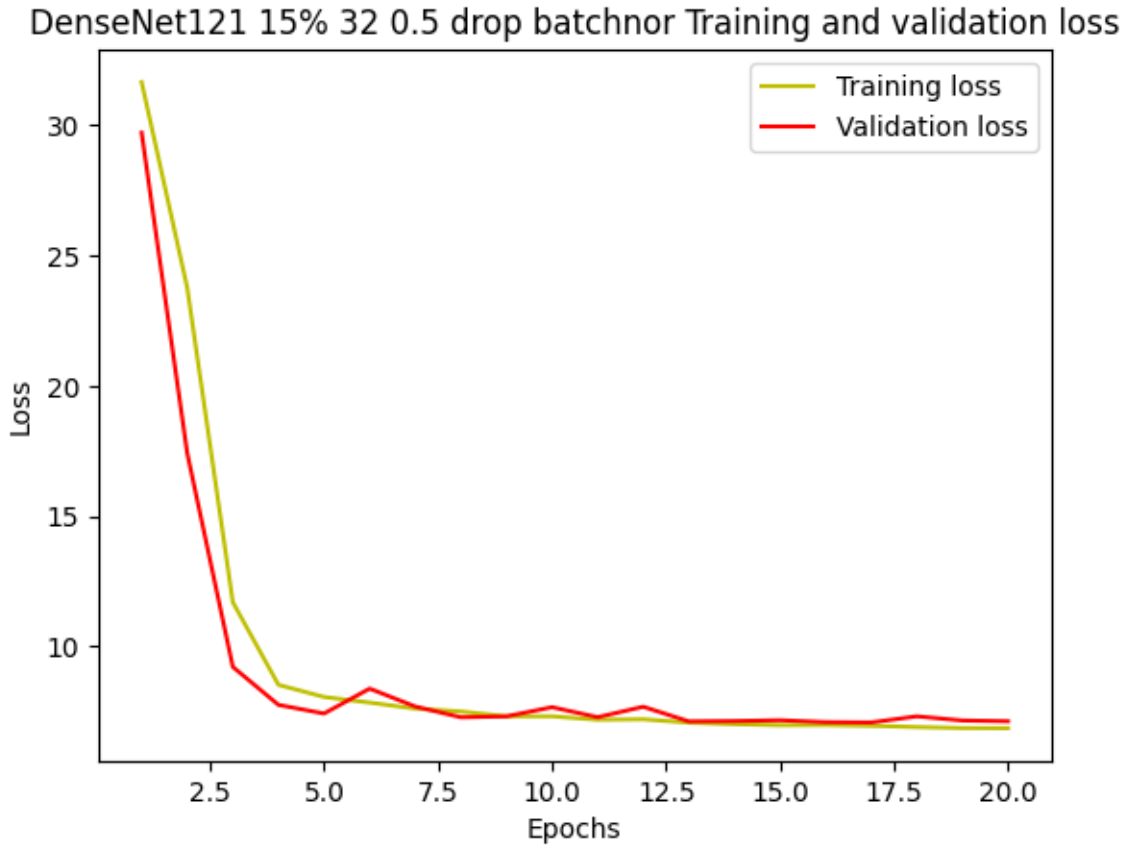
\

DenseNet121 (15% test)- 3 dense layers , 1 dropout rate 0.5 , shuffling , random state 42 , 1 batch normalization layer , batch 32 .

112/112 [==============================] - 3s 25ms/step - loss: 7.1343 - mae: 7.1343

Test Loss: 7.134298801422119

Test MAE: 7.134298801422119



Last version tables

Pre-trained	**DenseNet121**	**DenseNet121**	**DenseNet121**	**DenseNet121**	**DenseNet121**
Batch Normalization layers	1	1	1	1	2
Dense layers	3	3	3	3	5
Dropout layers(rate)	1(0.4)	1(0.3)	1(0.4)	1(0.5)	3(0.5)
Test split	20%	15%	15%	15%	15%
MAE score	7.88	7.14	7.17	7.32	underfit
Batch size	32	16	16	16	16

Pre-trained	**DenseNet121**	**DenseNet121**	**DenseNet121**
Batch Normalization layers	1	**1**	1
Dense layers	3	**3**	3
Dropout layers(rate)	1(0.3)	**1(0.4)**	1(0.5)
Test split	15%	**15%**	15%
MAE score	7.06	**6.99**	7.13
Batch size	32	**32**	32

The method I followed as we can see taking 3 dense layers , 1 dropout layer , 1 batch normalization layer with a split of 15% test and 85% training using **DenseNet121 for UTKface** and a random state of 42 shuffle and a shuffle while training and patience of 5 epoch for validation loss if not increasing and started to overfit to stop the training.

Lowest score achieved in my method: **6.99 MAE of regression**

Previous work results [2]

Table 2
Classification accuracy and MAE Regression in DCNN using UTK Face dataset

Technique	Classification accuracy			MAE Regression
	Individual	5 Years	10 Years	
GoogLeNet	0.78	0.92	0.88	9.17
DenseNet	0.86	0.91	0.89	9.34
ResNet	0.91	0.94	0.92	9.12

6. Conclusion

6.1 Benefits

a. Benefits to users :

1. Enhanced Age Prediction Accuracy : The implementation of deep learning models, specific Neural Networks (CNNs), has demonstrated promising results in accurately predicting age ranges.

2. Real world applications : using the age predication can be in various and different applications.

b. Benefits to me :

1. Going through the new technologies and discovering the power of computer science which this kind of project will get me some advanced skills within the new tools and skills in technology.

2. Learning and comparing the different CNN architectures and dive into these types of deep topics.

3. Improving and applying my understandings in machine learning and neural networks

4. Good startup for my master’s degree project

6.2 Ethics

Respect the privacy of those whose face data is utilized to train the model. Put policies in place to protect and anonymize personal data, get people's informed consent by explaining the usage of their data and the reason behind age detection.

Keep an effort to make the model's decision-making process transparent so that users understand how age detection are determined, recognize the limitations of age detection methods and inform users that the results are approximations and might not be exact.
Avoid stigmatization implement measures to prevent the stigmatization of individuals based on age predictions, emphasize positive use cases and benefits, community awareness engage with the community to understand concerns and perceptions related to age detection technology.

long-term Impact consider the potential long-term impact of age detection technology on

individuals and society, taking steps to mitigate negative consequences.

Legal Adherence: Comply with all applicable laws and rules when using facial recognition technology to determine an age. Keep updated of any changes to regulations in the area.

Why did I choose this project?

You should explain why did you choose these project, What was the main idea that attracted you to make this project (1-2 paragraph)

Taking this kind of project as my graduation project is rooted from my constant curiosity and eagerness to get through the latest advancements in my department and computer science .

This project presents a dynamic learning opportunity , allowing me to dive into the latest technologies and what I find out is choosing deep learning will be suitable for me as a start to these kind of deep topics in computer science .

As the technology evolves so this landscape and this project serves as a start platform for me to get some experience and develop my skills in deploying the deep learning art models for the real world applications .

exploring through the challenges of age detection through

deep learning models attracts my interest and fuels my motivation to learn by pushing myself into this project , I aspire witnessing the practical implication of age detection which will help me keeping up with the latest technological trends.

Pushing the latest techniques into my project and comparing it with different related works and results where I want to continue on this path for my master's degree and thesis.

6.3 Future Works

Going through this kind of projects desires with my interests and plans after graduation specially the topics which includes machine learning , big data , data science .

Age detection with deep learning project lays the foundation

for my future explorations.

Refinement skills of machine learning planning to undertake projects dives

deeper into advanced algorithims exploring big datasets.

Planning to take strong courses and startups in the company which taking this kind of project will help to get a Machine learning engineer job with company I took summer training with which I was suggested from the engineer who trained me in this field.

7. References

[1] : **Akhand, M. A. H., Ijaj Sayim, M., Roy, S., & Siddique, N. (2020). Human age prediction from facial image using transfer learning in deep convolutional neural networks. In Proceedings of International Joint Conference on Computational Intelligence: IJCCI 2019 (pp. 217-229). Springer Singapore.**

[2]:**Sathyavathi, S., & Baskaran, K. R. (2023). An intelligent human age prediction from face image framework based on deep learning algorithms. Information Technology and Control, 52(1), 245-257.**

- Bukar AM, Ugail H (2017) Automatic age estimation from facial profile view. IET Comput Vis 11(8):650–655. https://doi.org/10.1049/iet-cvi.2016.0486

- Eidinger E et al (2014) Age and gender estimation of unfiltered faces. IEEE Trans Inf Forensics Secur 9(12):2170–2179. https://doi.org/10.1109/TIFS.2014.2359646

- Fernández C et al (2015) A comparative evaluation of regression learning algorithms for facial age estimation. In: Lecture notes in computer science (including subseries Lecture notes in artificial intelligence and Lecture notes in bioinformatics), pp 133–144 (2015). https://doi.org/10.1007/978-3-319-13737-7_12

- Fu Y et al (2014) Interestingness prediction by robust learning to rank. In: Lecture notes in computer science (including subseries Lecture notes in artificial intelligence and Lecture notes in bioinformatics), pp 488–503. https://doi.org/10.1007/978-3-319-10605-2_32

- Geng X et al (2007) Automatic age estimation based on facial aging patterns. IEEE Trans Pattern Anal Mach Intell 29(12):2234–2240. https://doi.org/10.1109/TPAMI.2007.70733

Guo G et al (2009) Human age estimation using bio-inspired features. In: 2009 IEEE confer ence on computer vision and pattern recognition, pp 112–119. IEEE. https://doi.org/10.1109/CVPRW.2009.5206681

- Han H et al (2013) Age estimation from face images: Human vs. machine performance. In: 2013 international conference on biometrics (ICB), pp 1–8. IEEE. https://doi.org/10.1109/ICB.2013.6613022

-Huang G et al (2017) Densely connected convolutional networks. In: 2017 IEEE conference on computer vision and pattern recognition (CVPR), pp 2261–2269. IEEE. https://doi.org/10.1109/CVPR.2017.243.