Age Estimation and Web Application

COMP90055

Computing Project (25 Credits)

Name: Hong Zhang

Student Number: 901317

Name: Qijie Li

Student Number: 927249

Type of Project: Software Development Project

Supervisor: Prof. Richard Sinnott

Semester 1, 2019

AGE ESTIMATION AND WEB APPLICATION

COMP90055 - COMPUTING PROJECT (25 CREDITS)

Hong Zhang
School of Computing and Information System
University of Melbourne
Student Number: 901317
hongz7@student.unimelb.edu.au

Qijie Li
School of Computing and Information System
Universiyt of Melbourne
Student Number: 927249
qijiel@student.unimelb.edu.au

June 11, 2019

ABSTRACT

Age estimation is an important and classical task in computer vision. It has many real-word application such as access control. In the paper, we proposed two age estimation models namely, 8-classes age estimation model and 101-classes age estimation model. For 8-classes model, it assembles 3 trained classification models. It classifies the input image into 8 different age groups and the label accuracy is reported. Moreover, MAE is reported by transforming the 8-classes model into precise age estimator. For 101-classes model, it adapts pre-trained ResNet-50 and changes the output layer into 101 classes softmax and the output of it is a single predicted age. Additionally, MAE of it is also reported. Proposed models are trained on the dataset with different pre-processes settings. The parameters of the propose models are fine-tuned. The best result 8-classes model gets is 70% on the label accuracy and transformation to precise age estimation with MAE 5.6. The 101-classes gets MAE 4.8 for its best result. After developing age estimators, they are deployed to the website sever using flask, Nginx and uWSGI.

***K*eywords** Age Estimation · CNN · Web Application · Computer Vision

1 Introduction

The main goal of our project is to develop two age estimation models and deploy the developed models onto our website server. The first model is trying to classify the face images into correct age group, the other one is trying to predict the precise biological age of the person in the input image.

Age estimation still remains an active research topic in computer vision field these days. Age estimation could play an important role in many real-world applications. Age estimator could be applied to access control, which can prevent person who have not reach the age limitation to buy sensible products, such as tobacco and alcohol.[1] It could also help with the law enforcement by scanning the video records of suspects with age estimation. [2] Age estimation is a challenging and classical machine learning problem, even for human it is a challenging task. [3] Human's age not only depends on the personal factors of induvial, it also depends on the external environmental factors. [4] Consequently, People with same age could have different appearances. Due to the continuity of age, age estimation task is usually treated as a regression task. [1] In our project, we treated this task as a classification task, for which we tried to classify the face image into pre-defined age ranges. In term of precise age prediction model, we change the output classes into classes which only contains one specific age.

Convolutional neuron networks (CNNs) is a popular machine learning model which is widely used in image classification tasks. [5] It was first proposed by Yann LeCun et.al. (1998), called LeNet-5 to classify hand-written digits, which only contains 7 layers.[6] Due to the limitation of the computation power and lack of public available image dataset, the deeper network AlexNet [7] used to classify high resolution RGB images was devised until 2012. AlexNet is the winner of 2012 ImageNet Large Scale Visual Recognition Challenge (ILSVRC), which significantly outperformed models designed before by reducing the top-5 error on the ImageNet Validation dataset to 15.3%.[7] AlexNet has more than 60 million parameters and used Rectified Linear Units (ReLUs) to accelerate the training process. [7]Our proposed models for age estimation used more sophisticated CNNs, which have different architecture from the classical CNNs and achieved better performance than AlexNet.

Deployment of the CNN models to the website server is also a challenging task. Comparing to other machine learning models, CNNs are huge in term of the number of parameters. Usually, CNNs for image classification task with state-of-art performance could consist tens of millions of parameters. [8] Consequently, huge number of parameters makes the model size become huge and require large RAM space to store them. This feature leads to the difficulty to build web applications involves image classification models. In our project, we used Flask API warpped our model and deploy it to the webpage server.

2 Related works

Age estimation is an important machine learning task which lots of research has been done on this topic. There are two branches of age estimation task. One branch is trying to estimate the real biological age of a human from the facial image. The goal of another branch is to estimate the apparent age of a human. Varieties of publicly accessible datasets with human faces and age labels are created by the researchers to train and develop age estimation models.

DEX[2][1]

Deep Expectation (DEX) is an age estimation model propose by the Computer Vision Laboratory of ETH Zurich. It is the winner of The ChaLearn Looking At People (LAP) ICCV 2015 challenge apparent age estimation task. DEX also achieves the state-of-art results for real biological age estimation. The breakthrough idea they used in DEX is the novel regression formulation which calculates the expected values of the deep classification outputs. The DEX system takes advantage over VGG-16 trained on the ImageNet dataset. Re-training and fine-tuning are carried out to the pre-trained VGG-16 model. In their paper, they also mentioned a standard evaluation metric, Mean Absolute Error (MAE) for age estimation task. Additionally, the largest dataset with real age and gender label called IMDB-WIKI is proposed in their work.

Ordinal Regression with Multiple Output CNN[9]

Niu. Z et al. proposed a Multiple Output CNN model for the age estimation task. In their paper, they cast the age estimation as an ordinal regression problem due to the non-stationary property of aging process. CNN with multiple outputs is used to solve the binary classification sub-problems transformed from the original ordinal regression problem. Asian Face Age Dataset (AFAD) which contains 160K Asian face images and corresponding age annotation is proposed in their work. It achieves state-of-art result on the MORPH II dataset with MAE 3.27 and MAE 3.34 on the AFAD dataset.

ARN[10]

Agustsson.E et al. proposed Anchored Regression Networks (ARN) in their paper, which could be adapted to various regression tasks. In their paper, they test ARN on the real age estimation task. It achieved MAE 3.153 on the DEX IMDB-WIKI dataset which slightly outperformed DEX method. It assigns series of linear regressors to the soft partitioned input spaces with respect to the anchor points, which allows the ARN to learn the approximation. The validation of the ARN performance is carried out with the same pipeline as DEX, and the ARN regression layer is attached over the fully connected layer of VGG-16. It could achieve the state-of-art performance by only slightly modify the original network architecture.

SSR-Net[8]

A Compact Soft Stagewise Regression Network for Age Estimation Most age estimation models have state-of-art performance usually suffer from the bulky model size. SSR-Net is a compact model which only consume 0.32MB memory space, which also achieves the state-of-art performance under MAE metric. It could be easily adopted to the mobile devices due to the small memory usage. It takes advantage of DEX mentioned before. It reduces the number of neurons for the age classes by using a coarse-to-fine strategy. Specifically, it has 3 classification stages and each stage refines the previous stage's output. The quantization of ages is another problem that could cause by treading age estimation as a classification task. The dynamic range of each age group which could shift and scale depend on the input images is introduced to address this problem. There are other compact age estimation models such as MobileNet and DenseNet which have slightly lager size (around 1 MB). The MAE of MobileNet and DenseNet is 6.50 and 5.05 respectively. SSR-Net significantly outperformed them with MAE of 3.16.

3 Dataset

The dataset used in our project is a pubic human face images dataset called UTKFace. [11] UTKFace contains 23708 images and each image is label with age, gender, and ethnicity. The images' age labels cover a large age range from 0 to 116 years-old. The images in the dataset very in terms of pose, facial expression, illumination, occlusion, and resolution. However, the dataset is not evenly distributed. There are some ages have much more image samples within them, and there are relatively small number of images of people above 60 years-old. Beyond the original dataset, the pre-process of it was carried out.

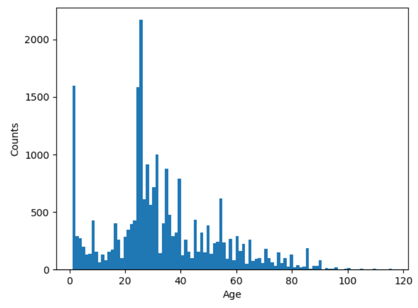

Figure 1: Dataset Overall Age Distribution

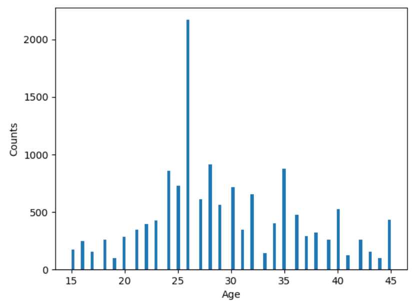

Figure 2: 15 to 45 Age Distribution

Image pre-processing

There are some images in the dataset which have exceeding background other than the person's faces. Those backgrounds provide noisy data to the training process, which could affect the performance of trained model. Hence, we need to perform some pre-process to the images in the dataset to remove those extra backgrounds in the images. Additionally, the human faces in those photos are not aligned uniformly. We also did facial alignment for the images in the dataset, which could potentially improve the accuracy of the trained model.

Face detection and landmark detection

Firstly, we carried out face detection to the images in the dataset. We used cascade classifier, which is available in the OpenCV-python[12] to detect and label the face area in the images. Next, based on the face detected, we performed facial landmark detection. Facial landmarks such as eyes, eyebrows, nose, and mouth could be used to do the face alignment. For landmark detection we also used a pre-trained model in OpenCV.

Image Cropping and Alignment

After detecting the face and landmark of the images, we cropped the images to the face area, and based on the landmark detected we normalized the images. For cropping, we set 3 different margin level, which are 0%, 20% and 40%. We set few checkpoints during the training process to determine the cropping margins which could potentially lead to the better model performance. The images cropped with different margins are stored into 3 different datasets for later experimental usage.

4 Model Review

There are two pre-trained image classification models used in our model, which are VGG-16 and ResNet-50. Both of the pre-trained models are available from the Keras package which are trained on the ImageNet dataset. In this section, these two models are reviewed.

VGG[13]

VGG is a deep CNN which makes improvements over AlexNet. It has 16-19 layers which is deeper than the AlexNet. It brings significant improvement to the performance of traditional CNN by increasing the depth of the networks

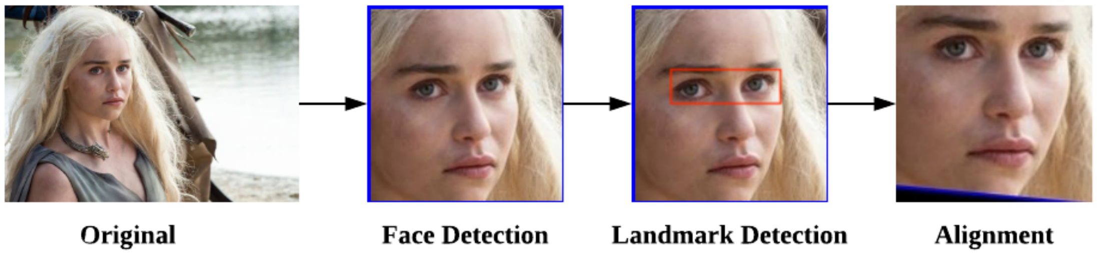

Figure 3: Image Pre-process

and changing the convolution filters into smaller size (3 x 3). VGG is the winner of ImageNet Challenge 2014 and significantly outperformed previous models. The input of the network is 224 x 224 RGB image and it is passed through a stack of convolution layers with 3 x 3 filters. The convolution stride is set to 1 pixel. There are 3 Fully Connected layers follow the convolutional layers, and all the hidden layers are using ReLU non-linearity. The total number of parameters is over 130 million, which make the model bulky in size. The model is trained on the very large ImageNet dataset, which contains approximately 1.3 million images. The best single model top-5 error on the ImageNet test validation dataset is 7.1% It suggests that the classical CNN architecture could achieve state-of-art performance of image classification task by substantially increasing the depth.

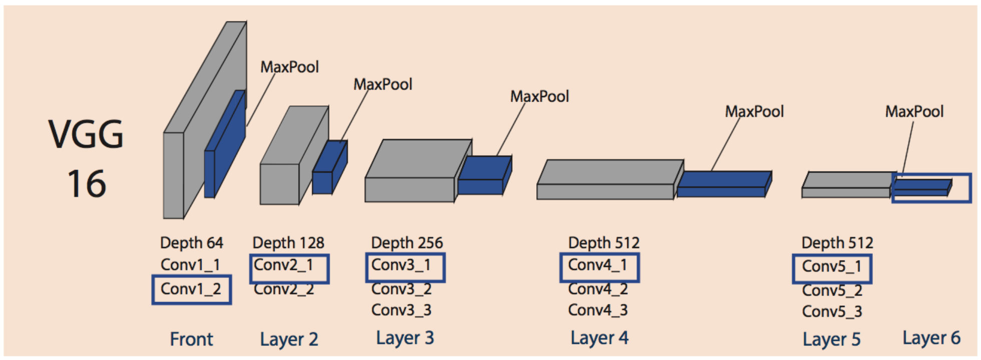

Figure 4: VGG Architecture

ResNet[14]

Deep Residual Net (ResNet) substantially increased the depth of deep convolutional neural networks by introducing the residual block which is the core feature of ResNet. With residual block the convolution neural network could go deeper without increasing the computational power required. One possible solution to increasing the depth is adding identity mapping layers, which leads to the deeper model could produce higher training error. Instead of inserting the identity mapping layers, ResNet addresses this by using the residual block, which creates shortcut connections between layers. It allows the ResNet could go up 152 layers and with lower complexity than the VGG. ResNet is the winner of ImageNet Challenge 2015, the ResNet with 152 layers achieves top-5 error 4.49% on the ImageNet validation dataset.

5 Experiment Setting

Based on the background knowledge disscussed above we proposed our own model and carried out the experiment with the dataset and different experimental settings.

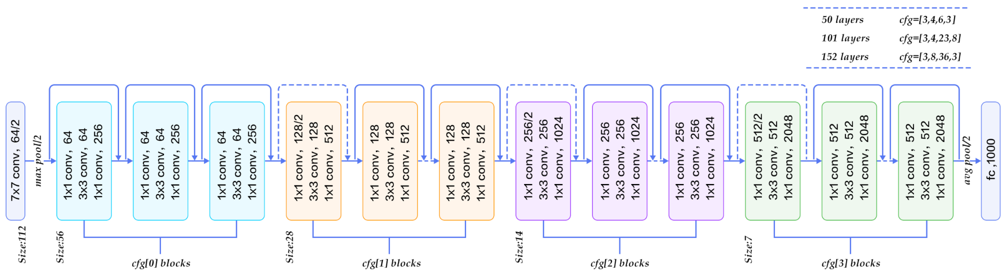
Figure 5: ResNet Architecture

5.1 Architecture Overview

We proposed two models with different age estimation outputs. The models described in this paper takes the advantage of pre-trained deep CNNs. Both models take the pre-processed image as their input, which is 224*224 pixels face RBG image. For the first model, it assembles outputs from 3 different models. The first output is from pre-trained VGG-16 network followed by one fully connected layer and 256 wide ReLU, the output layer is changed to 8-classes softmax. Other two models are fine-tuned ResNet-50. We added two fully connected layer to the pre-trained ResNet-50, one is with 521 wide ReLU and other one is 256 wide ReLU. The output layer is changed to 8-classes softmax as well. The difference between two fine-tuned ResNet-50 is that the parameters of the customized fully connected layers are trained on different margin level dataset. One is trained on 0% margin dataset, the other one is trained on 20% margin dataset. Finally, the three outputs are aggregated to give the probability of the image belongs to which age class. The precise age prediction model is built based on the ResNet-50 and change the output layer to 101 classes softmax.

5.2 Evaluation Metric

There are two model performance evaluation metrics used in our project.

MAE. Mean Absolute Error (MAE) is the standard age estimation evaluation metric which computes the absolute error between the predicted age and the age label of the image. This is used to evaluate the precise age estimation models proposed.

Label Accuracy. It is computed as the proportion of corrected labelled samples over number of all samples in the validation dataset. This is used to evaluate the 8-classes classification model.

5.3 System Design

The implementation details of the model proposed are talked about in this section.

5.3.1 Data Pre-process

For the 3 different margin level datasets, we did data cleaning and data augmentation to them.

Data Cleaning In term of data cleaning, we removed some images with obvious wrong label attached to them. For example, some baby's photos are labelled with 60 years-old and some elder people pictures are labeled with 10 years-old. During the cropping process, some images are cropped improperly, some cropped images do not contain persons' faces in them. We also removed those imags.

Data Augmentation As mentioned before, the dataset is not evenly distributed, which could cause the model trained is biased. To mitigate this problem, we adapted data augmentation technique to it. Before data augmentation in Keras generator[15], sample number of age classes with fewer samples are increased by random image augmentation and sample number of age classes with more images are ignored so that we partly equalize the age distribution. However, this can not be equally discriminative for all ages, thus data imbalance still exists in our dataset before the utilization of data augmentation in the Keras data generator which will be illustrated in Model training section stage 2.

5.3.2 Experiment Initialization

The initial setting of preprocessed dataset for model training and evaluation is that, the dataset is randomly splitted into 90% for training and 10% for testing during the process of creating database for dataset with different margin(0, 20%, 40%). Then the training set is splitted into 90% for parameter tuning and 10% for validation during the training phase in Keras backend with Tensorflow.

Two different networks for two different scenarios in our web application, one classification model for 8 output neurons and another for 101 neurons as output are trained. The former one is for age prediction for live-time video capture face detection and precise age estimation, while the latter one is parameter tuning are performed on both scenarios. The initial setting is a pre-trained VGG16 and ResNet50 with 224*224 preprocessed RGB image as input. The optimization is carried out using stochastic gradient descent method with mini-batches of size 64 and momentum value of 0.9 with weight decay set to 0.001. A dropout rate of 0.5 is used to regularize the network parameters during the training process. Normal distribution will be used to initialize the weights of the newly added fully connected layers with mean of 0 and standard deviation of 0.01, and bias are initialized to zero. The learning rate is set as 0.001 for the output layer. Then the output of each hidden layer is fed to the next hidden layer as in input to calculate log-loss softmax probability for each classes. The learning rate times 0.2 for every 10 epoches.

Our goal is to optimize and find the parameter of the fully connected layers that minimize the prediction of softmax-log-loss. The pre-trained convolution layers that are trained on ImageNet are freezed both for ResNet50 and VGG16 Nets and parameters are optimized for fully connected layers for this task. The initial network for ResNet is displayed in Figure 6 and VGG16 is displayed in Figure 7.

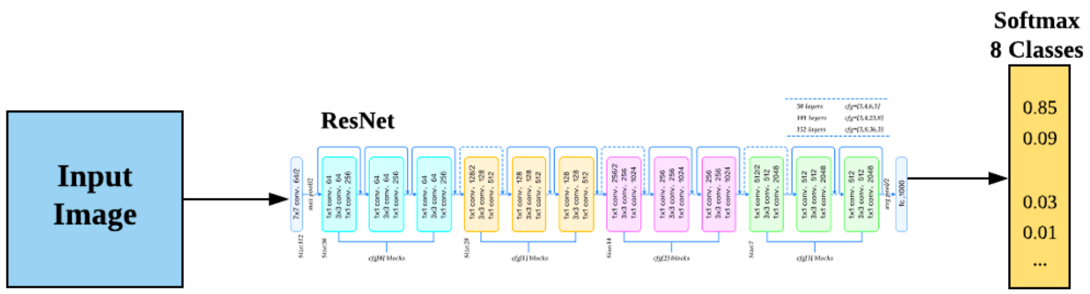

Figure 6: Initial ResNet

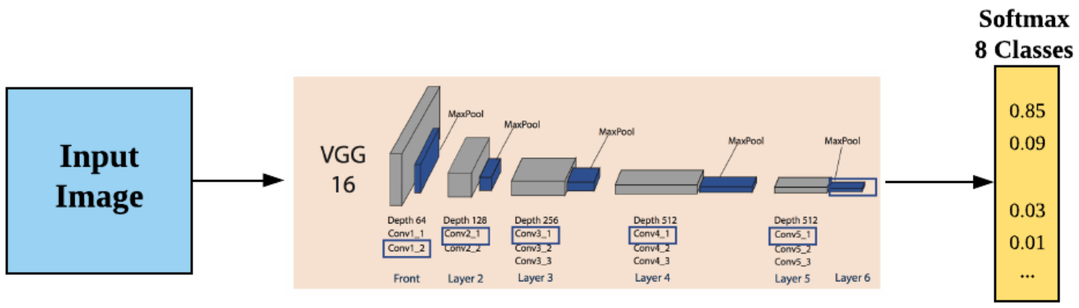

Figure 7: Initial VGG

5.3.3 Model Training

There are 3 stages of the whole model training process. In stage 1 the 8-classes classification model in trained. The 101-classes age estimation model is trained in stage 2. Stage 3 transforms the age classification model into precise age prediction model.

Stage 1: Networks training for 8 classes

Step 1: Model training and parameter tuning on both pre-trained CNN models
Start from the initialization setting, both VGG16 and ResNet50 models pre-trained on ImageNet are trained on training dataset of original margin dataset and 20 percent margin dataset and then test them on validation dataset for performance comparison. The following parameters are tuned on both models with options listed in the following Table 1.

Parameters	**Options**
Depth & Width of FC Layers	One 256-wide ReLu layer, 512-wide ReLU layer followed by 256 ReLU layer, 1024-wide ReLU layer followed by 512-wide ReLU layer followed by 256 ReLU layer
Input Pixels	119*119 pixels, 64*64 pixels and 224*224 pixels
Optimizer	Adam (lr=0.001) and SGD (lr=0.001, momentum=0.9) with learning rate step decay of 0.0005, lr times 0.2/ reduce 5 for each 10 epoches for convergence
Batch Size	32,64,128
Input Dataset Margin	Pre-processed images with 0, 20%,40% margin

Table 1: 8-Classes Model Parameter Setting

Step 2: Model ensemble

In our proposed networks, the final prediction is the average of the ensemble of classification probabilities from one VGG16 model, of which trained from dataset with no margin and 2 ResNet50 models trained from dataset with no margin and dataset with 20 percent margin. We take the average result of each output neuron from 3 different models to give a final predicted softmax probability result for each class/neuron. The final proposed ensemble networks architecture is displayed in Figure 8 with the fine-tuned parameters.

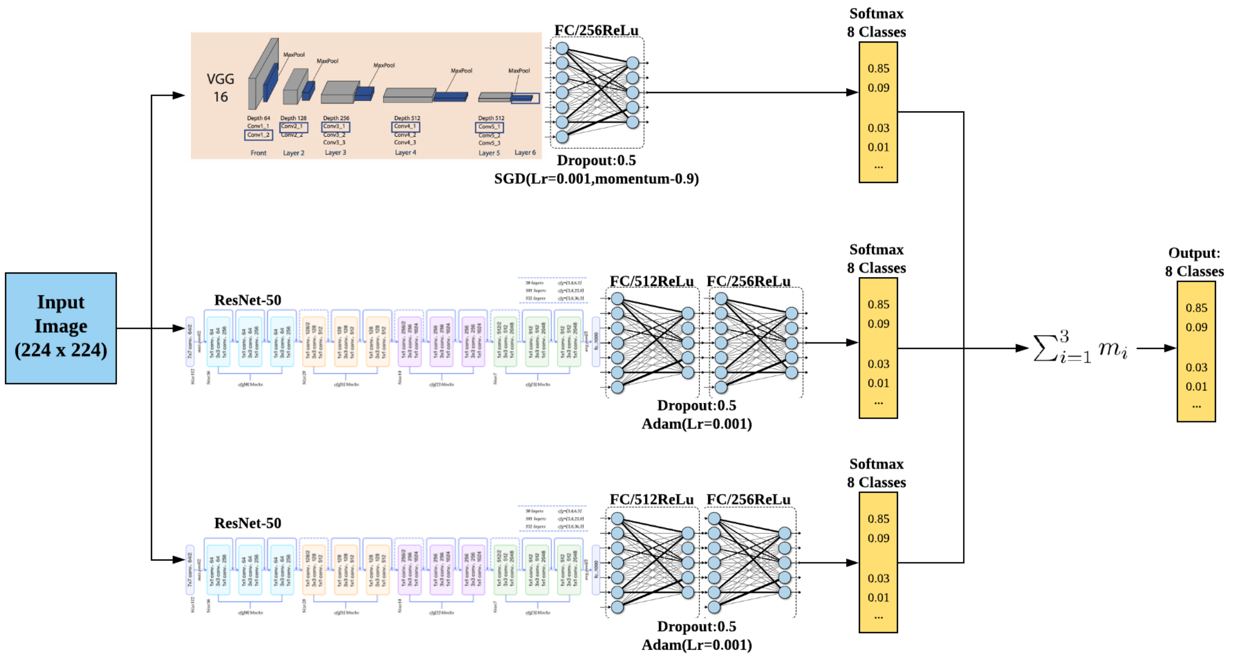

Figure 8: 8-Classes Model

Stage 2: Model training for 101 classes Precise Age Estimation

After building classification model for 8 classes, because the best accuracy result 0.69 from ResNet50 is better than 0.66 of VGG16 on the validation dataset so we decide to use Resnet50 in 101 age groups prediction. Model is trained to predict the precise biological age for 101 age classes use different classifier on top of the CNN learned features, because we think the age prediction problem can either be a regression or classification problem. On top of the CNN learned features, the reason we regard age estimation can be a regression one because age is a continuous feature as well as a discrete feature. In general, if the age prediction problem is defined as a classification problem, the age values are discretized into N ranges of age either by uniform age range or divided by equal numbers of dataset in each class so that data imbalance can be prevented. In this stage, there's no involvement of discretization into groups, each class is one age label. Thus, for the learning classifier, the following strategies are utilized with prediction performance of MAE value compared:

- Log-loss softmax probability, expected value calculation
- Learning a regression(SVR) on top of the CNN features of the previous layer
- Taking the age of the neuron with the highest probability

All the parameters we are tuning are listed in Table 2.
Moreover, the following techniques are utilized to prevent overfitting:

- **Regularizer and regularizer initialization**
  Different data augmentation techniques in terms of Keras ImageDataGenerator are utilized before fitting the model. This deals with the training sample imbalance and prevent overfitting by performing horizontal flipping, contrast, brighten, rotate, random zooming and distortion, etc. on images.

Parameters	Options
Pooling method	Global Average Pooling and Max Pooling for pre-trained CNN model output
input pixels	119*119 pixels, 64*64 pixels and 224*224 pixels
optimizer	Adam (lr=0.001) and SGD (lr=0.001, momentum=0.9) with learning rate step decay of 0.0005, lr times 0.2/ reduce 5 for each 10 epoches for convergence
image margin	0, 20%,40% margin
batch size	32,64,128
learning strategy	Regression, Argmax classification and Expected Value

Table 2: 101-Classes Model Parameter Setting

- **Data augmentation**Different data augmentation techniques in terms of Keras ImageDataGenerator are utilized before fitting the model. This deals with the training sample imbalance and prevent overfitting by performing horizontal flipping, contrast, brighten, rotate, random zooming and distortion, etc. on images.
- **Label augmentation**
  Label Augmentation is to add Gaussian noise to the age labels by adding age with random Gaussian value instead of the transformed one hot vector of 101 original age labels. This technique promotes performance on prediction MAE on validation data and is proven to greatly prevent model overfitting.
- **Random erasing**
  Random Erasing is one way of image augmentation methods just like image flipping, etc. for image preprocessing. This method tries to regularize models using training images that are randomly masked with random values. The probability that random erasing is applied, minimum or maximum proportion of erased area against input image, minimum or maximum aspect ratio of erased area can be defined as input and we are utilizing an existing class of random erasing.
- **Mix-up generator are applied during data training using keras fit_generator**
  Mix-up is a kind of image augmentation methods, which augments training data by mixing-up both of training images and training labels by linear interpolation with weight lambda:
  $$x = lambda \times x_1 + (1 - lambda) \times x_2$$
  $$y = lambda \times y_1 + (1 - lambda) \times y_2$$
  where x is the training images and y is the training label lambda is drawn from the Beta distribution Be(alpha, alpha).

Stage 3: Transformation of classification to precise age estimation

In this part, a transformation from the 8 classes classification problem into a precise age prediction problem is performed. As illustrated in the 101 precise age prediction model, the model on top of the pre-trained CNN model is trained for both regression and classification because age can be a continuous value as well as discrete classes. So we did the same experiments in this part using 3 different learning strategies. We compare the MAE results of the model transformed from classification model of 8 classes with 101 ages precise model across all the varying learning strategies.

In the transformation, for regression we replace the last layer with only 1 output neuron encoded from input of the last layer with 8 neutrons. And in case of learning strategy of expected value, the following formula is followed to calculate a precise age prediction value:

$$E(O) = \sum_{i=1}^{|Y|} y_i \cdot o_i$$

where Y is the number of dimensional output neuron, O is softmax probability of each age class. y is Relative frequency distribution/probability mass of sample ages in each age group range. Lastly, as in case of classification learning strategy, the argmax value of the age class is chosen and the precise prediction result is y of this age class.

Performance comparisons will be made for single models and ensemble models. Analysis will be given on how the transformation of 8 neurons output prediction of precise age performs compared with the direct 101 precise age prediction in part 6.1.2.

5.4 Implementation Details

For our hardware settings, we use a single nvidia GTX 1070 graphics card, i7-7700HQ CPU, 32GB RAM, and 2TB hard disk drive. The training time for the Residual Networks with 512-wide ReLU layer followed by 256 ReLU layer architecture is around 26 hours for 40 epochs for 8 classes classification. Fine-tuning of parameters may take couple of hours depends on the parameters being tuned. The whole 101 classes Residual networks takes 2.5 days to train for 40 epoches on around 20 thousand training dataset.

6 Experiment Result and Analysis

The experiment results of precise age estimation model and age range estimation model are disscussed and analyzed separately.

6.1 Precise Age Estimation Results Analysis

The results of 101-classes model and the transformation of age range to precise age are presented here.

6.1.1 Results and Improvements on 101 classes model

In this section the performance of the proposed 101 prediction model for estimating the real (biological) age is illustrated. The techniques used to prevent overfitting, including data augmentation, label augmentation, random erasing, mix-up generator applied during data training using keras fit_generator, etc. greatly promotes the model MAE on validation dataset.

The proposed model with best performance evaluated by MAE is the model using global average pooling for ResNet50 pre-trained model output, optimizer of Adam (lr=0.001) with learning rate times 0.2 for each 10 epoches for model convergence. In aspects of input preprocessed images, the image input pixels of 119*119, with 20% margin performs the best when taking the context around the face into consideration as model input for feature generation. The reason behind the result is that if the face is too large, it will cause a problem of no context existing on some of the sides, in other word, the last pixel at the border is a pure repeat of other pixels. The inclusion of cropping with margin can ensure that the face is always located in the same area for all the preprocessed face images. Moreover, actually the other image preprocessing steps: face detection with face alignment including the eyes with a horizontal strip promotes the performance a lot as well because eyes are essential features for age prediction in the human face. Moreover, images with RGB perform better than grey-colored images.

Figure 9, Figure 10 and Table 3 show the results and improvements of performance from model training based on no augmented data, to model trained with data augmentation only (including horizontal flipping, contrast, brighten, rotate, random zooming and distortion, etc. on images), to model that take label augmentation into consideration in the generator of augmentation, also with an automatic learning scheduler in the model training process that learns the best optimizer and learning rate of the model. The final proposed 101 class model is applied on a mix up generator in Keras for training and achieves the lowest MAE on validation dataset of 4.8 and loss of 2.9. Figure 10 demonstate the performance of proposed model on both training and testing dataset. From Table 3 we concluded that Label Augmentation performs a cardinal role in promoting model performance and achieve remarkable improvements in reducing MAE. By simply incorporating label augmentation into model can reduce MAE for nearly 1 year range, which is up to a 17.2% reduction on MAE.

Ways of overfitting prevention	**MAE on validation dataset**	**Loss onvalidation dataset**
Image preprocessing with no Augmentation	6.8	4.1
Data Augmentation only	5.8	3.7
Mixup Generator	5.6	3.5
Label Augmentation & Optimizer and LR tuning	4.8	2.9

Table 3: Performance using different ways of overfitting prevention

6.1.2 Transformation from Age range to precise age prediction

As introduced before, a transformation is performed from 8 classes classification result into a biological precise age based on Relative frequency distribution/probability mass of sample ages in each age group range.

The performance evaluations in Table 4 reveal the disparity between the combinations of different settings: number of output neurons (8 and 101 classes in this project) and different pre-trained model with different learning method applied for parameter learning and optimization on top of the pre-trained model. For all the combinations of settings, we use the 20% margin training dataset for model training and validation.

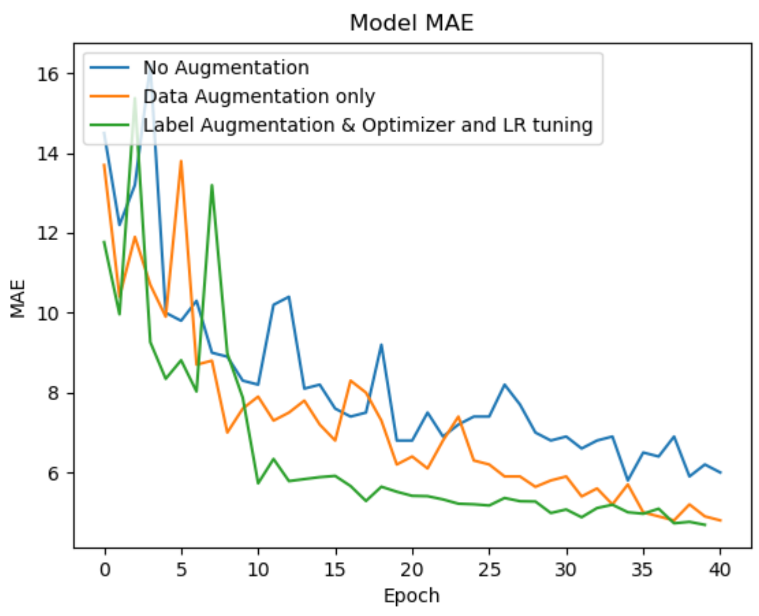

Figure 9: Comparison of training process

Pre-trained Model	Number of Output Neutrons	Learning Strategy	Best Accuracy	MAE
ResNet50	8	Expected Valued	0.69	5.7
VGG16	8	Expected Valued	0.66	5.9
Ensemble	**8**	**Expected Valued**	**0.70**	**5.6**
ResNet50	**101**	**Expected Valued**		**4.8**
ResNet50	101	Classification(Argmax)		5.6
ResNet50	101	Regression		5.5
ResNet50	8	Classification(Argmax)	0.64	5.9
ResNet50	8	Regression	0.61	6.2

Table 4: Model Comparisons

The choice of learning method on top of the pre-trained model makes differences on the prediction performance in a large extent.

For a choice of regression learning method, the outliers cause large error leading to unreliable and large gradients makes the model convergence to be hard, thus leading to easily overfitting problem and the unreliable and unstable predictions.

As for a log-loss softmax/expected value learning method, it can promote the robustness during the training stage and accuracy during testing stage. Moreover, it provides additional benefits of providing interpretations of the output probability distribution to estimate the confidence for each neuron/class prediction.

Table 4 also demonstrates that 101 classes with expected value learning method outperform 8 classes model. The reason explains the low MAE of 8-classes classification model is that the training and validation dataset owns different distribution of ages in each age range, thus the probability mass is only a reflection of the training data itself and actually are weights that do not make sense for either validation dataset or other images we would make predictions on. Future parameter learning on y value learning for each group range will definitely improve the prediction performance.

6.2 Age range prediction model results analysis

The proposed 8 class networks with best performance evaluated by accuracy is the ensemble model of one VGG16 model, of which trained from dataset with no margin and 2 ResNet50 models trained from dataset with no margin and dataset with 20 percent margin. The ResNet50 using global average pooling for ResNet50 pre-trained model output, optimizer of Adam (lr=0.001) with learning rate times 0.2/reduce 5 for each 10 epoches for model convergence. In

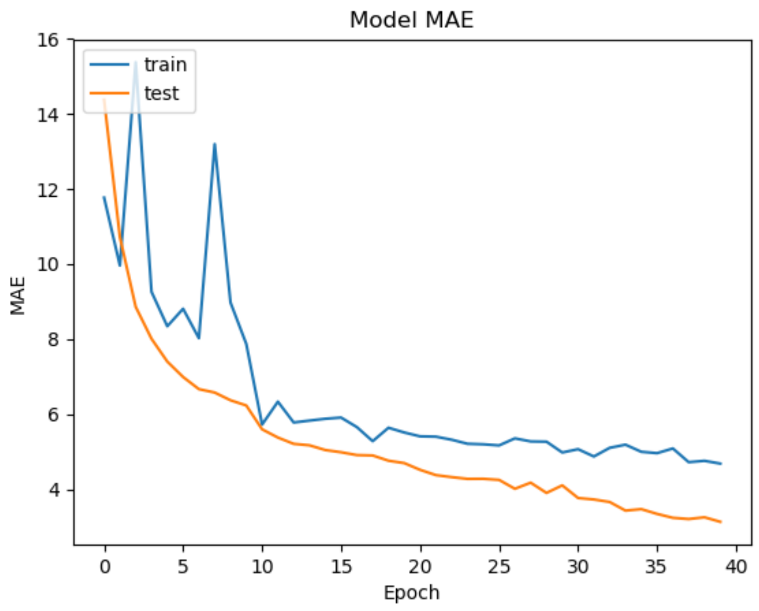

Figure 10: Performance on training and test dataset

aspects of input preprocessed images, the image input pixels of 119*119. Figure 11 demonstrate the disparity between different models with the most optimized parameters.

The fully-connected layer structure of Resnet50 and VGG16 are proposed differently. The Residual nets has a fully connected layer structure of a 512-wide ReLu layer followed by a 256-wide ReLU layer. The advantage of ReLU over sigmoid is that it trains much faster because the derivatives of sigmoid becomes very small is the saturating region and therefor the updates to the weights almost vanish. Furthermore, by using the dropout rate can reduce overfitting problem by randomly switching off the activation with probability of 0.5 and provided structured model regularization.

Ensemble Model achieves best accuracy of 0.70, the best ResNet50 achieves accuracy of 0.69, followed by VGG16 of accuracy of 0.66 compared to the initial model of only 0.48. Figure 12 and Figure 13 demonstrate the performance in accuracy and loss on training dataset in each epoch during training.

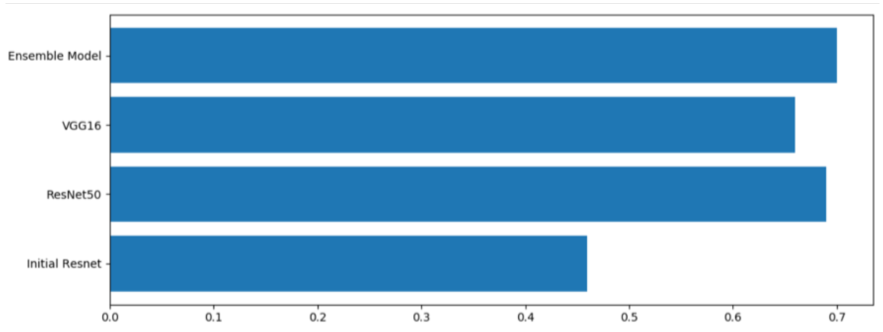

Figure 11: Model Accuracy Comparison

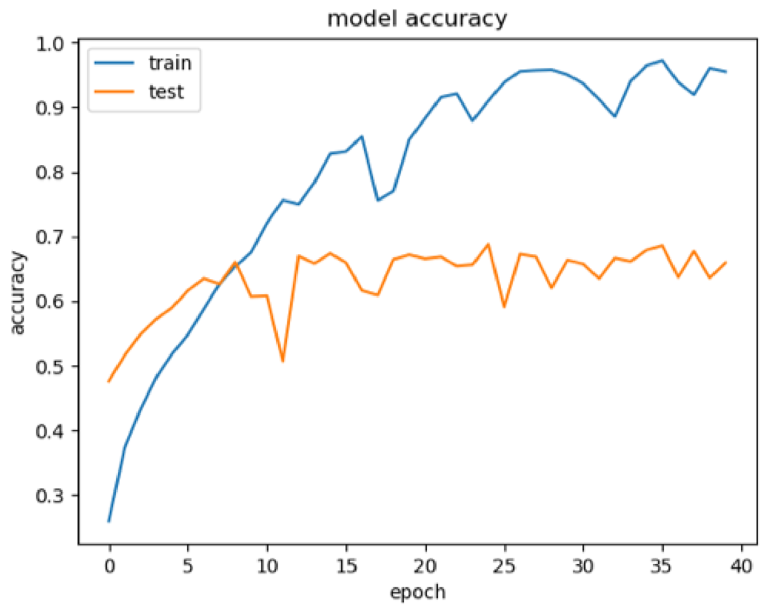

Figure 12: Model Accuracy

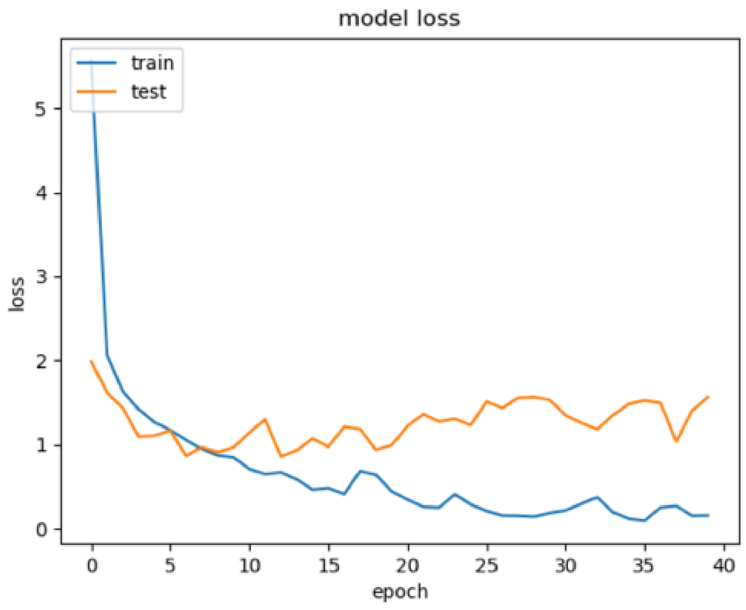

Figure 13: Model Loss

6.3 Prediction Analysis

In this section, confusion matrix is displayed both in 8 classes and 12 classes on our testing data. Figure 14 and Figure 16 shows the confusion matrix drawn from ResNet50 and ensemble model. The confusion matrix can reveal how the model performs in each age group. As the one for our proposed ensemble model displayed in Figure 15, age group from 0-2, 2-6, 7-14, 24-34 performs relatively good, at least correctly predicting the age group for 70%. Age group prediction between 35-44 and 44-59 is relatively worse, with the age range 35-44 below 50%. We also find that there's more probability to give older age prediction than younger ones. The possibility behind that could be the apparent images in the database tends to be younger than the actual biological age.

Another conclusion can be made by the comparisons between two different model is that different model utilize different features for classification, thus leading to different good performance age range of the model itself. For example the ResNet50 make a better prediction in age range 7-14, while VGG16 make better prediction in age range 24-34. The ensemble of models is proven to be able to prevent overfitting by balancing the predictions and lead to better overall accuracy on testing data.

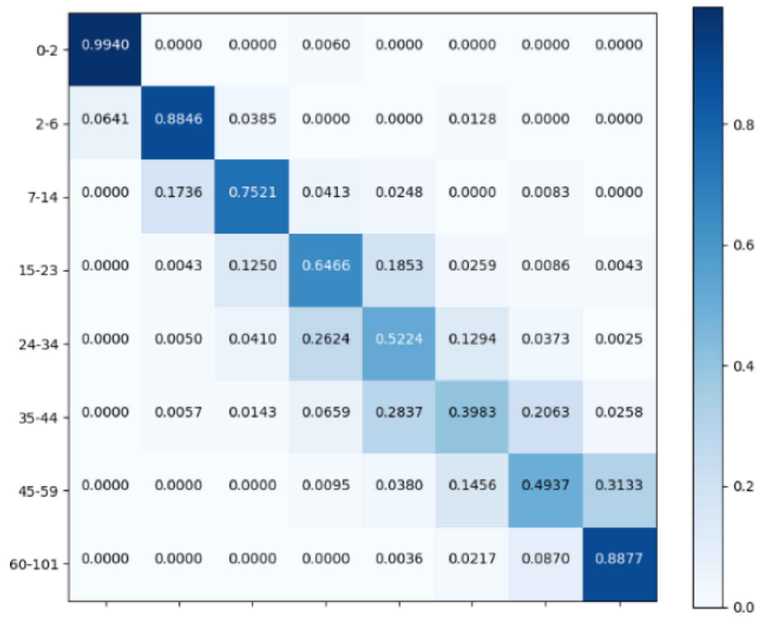

Figure 14: 8-classes Model Confusion Matrix

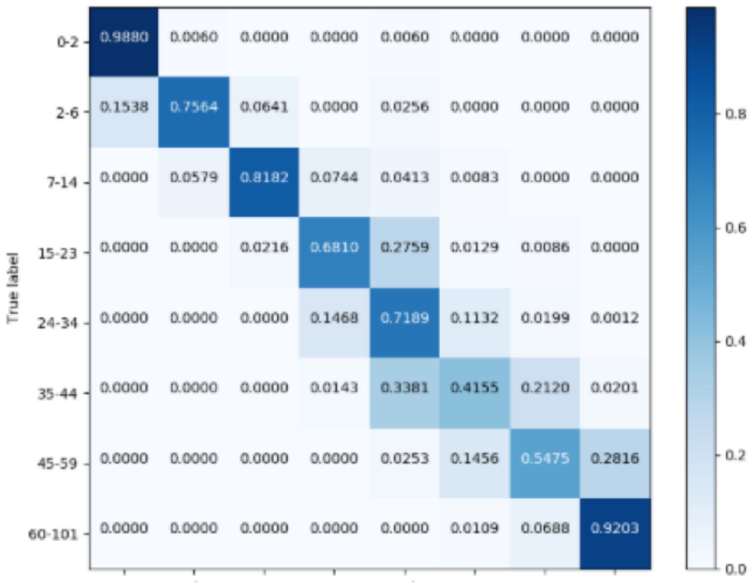

Figure 15: ensembled 8-classes Model Confusion Matrix

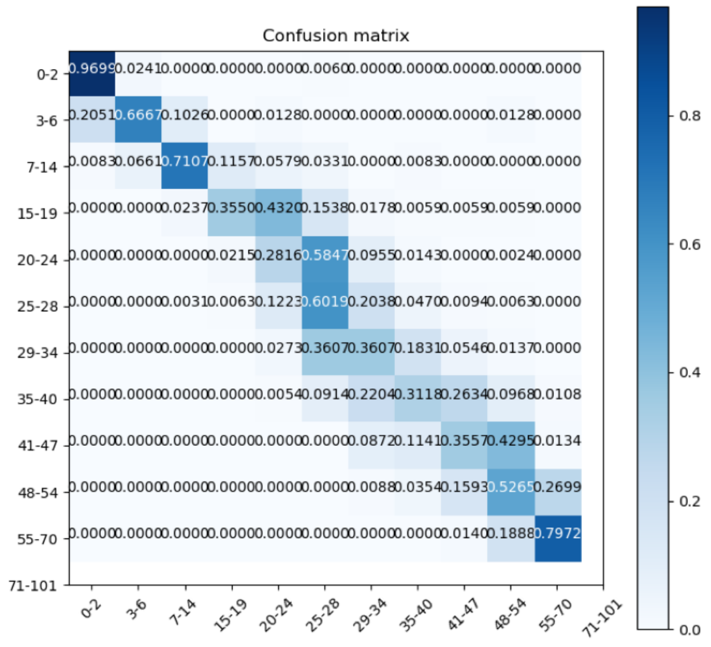

Figure 16: Confusion Matrix drill-down into 12 classes range.

6.4 Prediction Example

Figure 17 is the example image of Robert Downey of biological age 21. We take this image as our input and get the cropped output with a prediction of 23 years old.

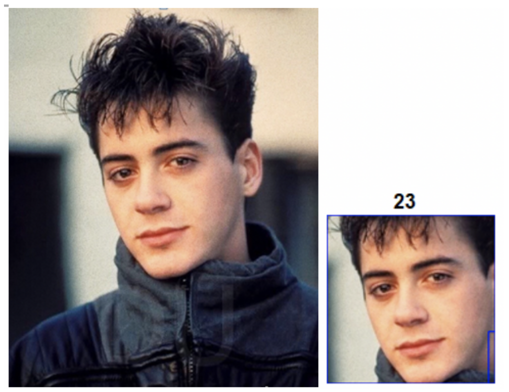

Figure 17: Example Output

7 Web Application Development and Deployment

After developing the age estimation models we deployed them onto our website sever. The web development details are discussed here.

7.1 Web Development

The 101 classes classification model is applied on live streaming prediction through webcam face detection. The 8 classes classification is applied on photo uploading or photo taking, displayed as pie chart for probability given each age ranges.

7.1.1 Web Application Framework

Introduction to Flask
Python-based Flask[16] web framework is utilized to build an API for our web application. Flask is a micro web framework written in Python and is considered much more lightweight. Flask is considered more Pythonic than Django because Flask web application code is in most cases more explicit and rapid high-level application development that offers useful libraries for the web applications.

How Flask Framework work in project
The Flask Framework looks for HTML files in a folder called static and gets .html page in the folder based on different application routes. A flask application instance is created by app = Flask(__name__,static_url_path=”). Web browsers send requests to this flask application instance, for each URL requests, the flask instance keeps a mapping from URL to a function we defined for each route. In our application, most of the functions are simple ones such as app.route(’/classify_system.html’) for 8 classes age range classification prediction, app.route(’/video.html’) for live-streaming video capture, app.route(’/about’) and app.route(’/homel’), which are utilized internally to send static files from the static folder to the browser. Additional, for our live-streaming video capture function, flask server can easily return Response by generating a video camera class we build by ourselves and return the face-detected and cropped image with frame and precise age displayed back to the users who send the request.

7.1.2 Web System Architecture

The web development server that comes bundled with Flask is not robust, secure, or efficient enough to work in a production environment. In order to make our web system more robust and efficient, we have to implement suitable web system architecture. Our goal is to achieve high responsiveness under load, good performance, high scalability of the system, as well as security ensured. Thus, we take a bundle of nginx[17], uwsgi[18] and flask as system architecture in which nginx supports reverse proxy to handle requests and pass back responses for Python WSGI servers.

Nginx
Nginx is a web server which can also be used as a reverse proxy, load balancer, and HTTP cache. It helps to manage all incoming traffic and distributes it to slower upstream servers. Nginx can help promote the speed and responsiveness of processing all our static files since we have lots of static files. Additionally, Nginx can automatically perform indexing and served as the load balancer.

uWsgi
uWSGI is an application server commonly used for Python applications. It is a contract between Flask and our web servers (UWSGI). A Web Server Gateway Interface (WSGI) server implements the web server side of the WSGI interface for running Python web applications. WSGI acts as a standard interface that modules and containers could implement. The web server is configured to pass the response in the form of HTML back to the requester. Such deployment architecture is scalable, cost-efficient and portable in nature.

Structure of web application files
Flask is used to send static files from the static folder to the browser. The Dockerfile, FlaskServer.py and requirement.txt should be in the same layer in virtual environment.

7.2 Deployment with Docker

Python-based Flask web framework is utilized to build an API for our machine learning models and flask application is containerized neatly inside a Docker image for deployment. Docker lends itself naturally to this problem as all the dependencies of the application can be packaged inside a container and scalability can be achieved by simply deploying the containers when the situation demands it.

In order to allow Docker to host our API, we need to specify a set of instructions that allow Docker to build the image. This set of instructions are stored inside the Dockerfile, which is used to build docker image and then executed to become a docker container.Our goal is to speed up the process of getting development and production environments up and to get a Flask app running on a production web server (not with Flask's app.run built-in development server) so that it is running in a portable way.

In the Dockerfile, Docker is instructed with what image to pull from the Docker hub repository. Image of tiangolo is used, which has python, flask, nginx, and uWSGI (the bridge between Flask and Nginx) as well as some other tools like git installed for us on a Debian OS. Since a virtual environment is required when deploying the Flask web server we need to set up a virtual environment before running our flask server and starting building our docker image: webapp: latest. The virtual environment with source venv/bin/activate is activated. Daemonization is performed because our app is desired to run in the background and to restart automatically and persistently if our server machine is restarted.

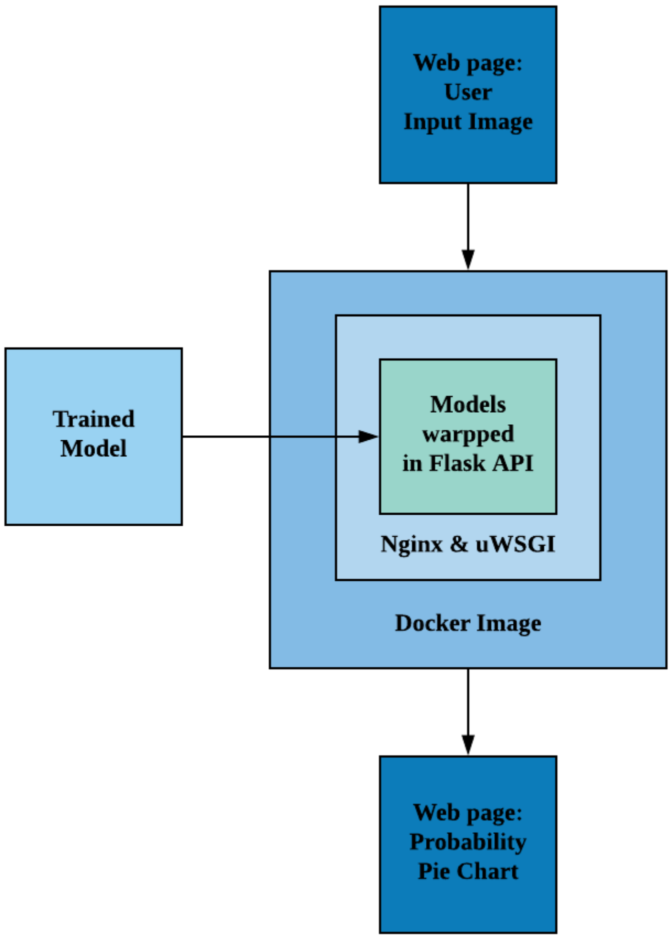

Figure 18: Web System Architecture

8 Conclusions and Limitations

In conclusion, We trained two age estimation model using the UTKFace dataset, namely 8-classes age range estimator and 101-classes precise age estimator. Moreover, the 8-classes estimator is transformed to precise age estimator. The performances of the models are showed and analyzed. After developing the age estimation models, they are deployed to the website server using Flask wrapper, Nginx and uWSGI. The leading limitation is that the images in the dataset are not evenly distributed with respect to the age label which leads to the bias in the trained model. Moreover, the model size is relatively large which cause the run time of the model on the website server is long. To mitigate this problem, we could compress the model to make the web server run faster. The static age range is used in our model which could be replaced by the dynamic age range method. Finally, the users' feedback about weather the prediction result is accurate or not could be collected from the website to give the model further improvement.

9 Appendix

Go to our website: `www.4everyoung.group:5000` website or run locally: python server.py and go to http://127.0.0.1:5000 to get access to our website.
Web Application Demo: `https://youtu.be/kNEIdHHt_Yg`
Project Code: `https://github.com/cherylzh0110/AgeEstimationApp`

References

[1] Rothe Rasmus, Timofte Radu, and Luc Van Gool. Deep expectation of real and apparent age from a single image without facial landmarks. *Computer Vision Laboratory, ETH Zurich*, 2016.

[2] Rothe Rasmus, Timofte Radu, and Luc Van Gool. Dex: Deep expectation of apparent age from a single image. In *International Conference on Computer Vision Workshops*, pages 252–257. IEEE, 2014.

[3] Han Hu, Otto Charles, and K. Jain Anil. Age estimation from face images: Human vs. machine performance. In *The 6th IAPR International Conference on Biometrics (ICB)*, pages 1148–1161. IEEE, 2014.

[4] Zhang Yunxuan, Liu Li, Li Cheng, and Loy Chen Change. Quantifying facial age by posterior of age comparisons. *arXiv preprint arXiv:1708.09687v2*, 2017.

[5] Yang Xulei, Zeng Zeng, Teo Sin G., Wang Li, Chandrasekhar Vijay, and Hoi Steven. Deep learning for practical image recognition: Case study on kaggle competitions. In *24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 923–931, 2018.

[6] LeCun Yann, Leon Bottou, Bengio Yoshua, and Haffner Patrick. Gradient based learning applied to document recognition. IEEE, 1998.

[7] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. *Neural Information Processing Systems*, 25, 2012.

[8] Yang Tsun-Yi, Huang Yi-Hsuan, Lin Yen-Yu, Hsiu Pi-Cheng, and Chuang Yung-Yu. Ssr-net: A compact soft stagewise regression network for age estimation. In *Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI-18)*, pages 1079–1084, 2018.

[9] Niu Zhenxing, Zhou Mo, Wang Le, Gao Xinbo, and Hua Gang. Ordinal regression with multiple output cnn for age estimation. In *Conference on Computer Vision and Pattern Recognition*, pages 4920–4928. IEEE, 2016.

[10] Agustsson Eirikur, Timofte Radu, and Luc Van Gool. Anchored regression networks applied to age estimation and super resolution. In *International Conference on Computer Vision*, pages 1652–1661. IEEE, 2017.

[11] UTKFace. `https://susanqq.github.io/UTKFace/`. [Online].

[12] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[13] Simonyan Karen and Zisserman Andrew. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations (ICLR)*, 2015.

[14] He Kaiming, Zhang Xiangyu, Ren Shaoqing, and Sun Jian. Deep residual learning for image recognition. *Microsoft Research*, 2015.

[15] François Chollet et al. Keras. `https://github.com/fchollet/keras`, 2015.

[16] Flask. `http://flask.pocoo.org/docs/1.0/`. [Online].

[17] Nginx. `https://www.nginx.com`. [Online].

[18] uWSGI. `https://uwsgi-docs.readthedocs.io/en/latest/index.html`. [Online].