Age Prediction on IMDB/WIKI face images using Deep Learning

Objective

The main objective of the project is to predict the age of celebrities based on their IMDB/WIKI face dataset. Some other objectives includes

- Design and implement different convolutional neural network (CNN) architectures using PyTorch.
- Use IMDB/WIKI dataset (faces only) for training, testing, and validation of network.
- Evaluate the performance of designed neural networks.

The project execution involved pre-processing raw data, implementing different CNN architectures, training and testing on IMDB/WIKI dataset and obtain accuracy results on age prediction.

Processing Raw Data:

Raw data from IMDB-WIKI - 500k+ face images with age and gender labels (ethz.ch) contains the images of faces along with the metadata containing the information as date of birth, the year when the photo was taken, the path of the file, gender and others. The data was processed as:

1. **Age Labelling and Outlier Filtering:** Labelled age was extracted from the metadata using date of birth and the time the photo was taken. The script *imdbwiki_age.m* extracts necessary information and generates tabular dataset *imdb_wiki.csv*. Outliers with ages less than 10 and ages greater than 100 are filtered using *Data_filtering.ipynb* script. Both age and image values are normalized to get the values between 0 and 1. Any large image sizes greater than 64x64 pixels are also filtered.

2. **Data preprocessing and splitting:** Images are converted to grayscale in *data_preprocessing.ipynb* script and the data is split in *data_splitting.ipynb* into training (60%), validation(20%) and testing(20%) data with 60, 20, 20% as shown in Table 1.

Table 1: IMDB_WIKI data size in after processing and splitting

IMDB_WIKI data	Data after filtering age and size	Training Data	Testing data	Validation Data
523,051	507,747	304,648	101,550	101,549

Learning Architecture

Three different CNN architectures as shown in Table 2 are designed and implemented for the age prediction. All architectures take the grayscale input image of size 64x64.

1. **Network architecture 1**
   A simple variant of VGG-16 CNN architecture is implemented with two convolutional layers followed by two dense linear layers. A unit size stride along with dropout is implemented. The age is predicted using regression model. Adam optimizer with learning rate 0.0005 has been used to minimize the mean square loss.

2. **Network architecture 2**
   This architecture is close to architecture 1 with two convolutional layer and one linear layer. Age is predicted using classification model. A different kernel size of 2x2 with stride (2,2) with maxPool2D kernel size 2 and stride 2, without dropout is used. Adam optimizer with learning rate 0.07 has been used to minimize the cross-entropy loss.

3. **Network architecture 3**
   This network architecture is similar to the architecture 2 except change in kernel size from 2x2 to 3x3 and slightly less dense linear network layer at the end.

	Architecture 1	**Architecture 2**	**Architecture 3**
Prediction model	Regression	Classification	Classification
CNN Architecture	**CNN layer 1** o Conv-64,kernel 5X5 o BatchNorm o Relu o Maxpool:kernel 2X2 o Dropout(0.5) **CNN layer 2** o Conv-128,kernel 5X5 o BatchNorm o Relu o Maxpool:kernel 2X2 o Dropout(0.5) **Linear layer 1** o linear(21632, 4096) o Relu **Linear layer 2** o linear(4096,1)	**CNN layer 1** o Conv2d(1,4,kernel_size=(2,2), stride=(2, 2)) o BatchNorm o ReLU o MaxPool2d(kernel_size=2,stride=2) **CNN layer 2** o Conv2d(4,4,kernel_size=(2, 2), stride=(2, 2)) o BatchNorm o ReLU o MaxPool2d(kernel_size=2,stride=2) **Linear layer 1** Linear(in_features=1024, out_features=100)	**CNN layer 1** o Conv2d(1,4, kernel_size=(3,3), stride=(2, 2)) o BatchNorm o ReLU o MaxPool2d(kernel_size=3,stride=2) **CNN layer 2** o Conv2d(4,4,kernel_size=(2,2), stride=(2, 2)) o BatchNorm o ReLU o MaxPool2d(kernel_size=2,stride=2) **Linear layer 1** Linear(in_features=900,out_features=89)
Loss function	MSE	Cross Entropy	Cross Entropy
Optimizer	Adam	Adam	Adam
Learning rate	0.0005	0.07	0.07
Batch size	100	100	100
Max. Epoch	25	25	25
Convergence tolerance	0.00001	0.00001	0.00001
Training, validation, Testing dataset	80K, 10K, and 10K	80K, 10K, and 10K	80K, 10K, and 10K
Library	PyTorch with CPU + GPU	PyTorch with CPU + GPU	PyTorch with CPU + GPU
Platform	Kaggle	Kaggle	Kaggle

Results

Training validity is shown in Fig. 1 and the age predictions for small size data points and relatively bigger size data points are shown in Fig. 2 and Fig. 3. The accuracy is computed with the threshold of 5 i.e. for the true age of 30 if the predicted age falls between 25-35, it is assumed to be correct. The accuracy for architectures 1, 2, and 3 were found to be 76, 70 and 71 %.

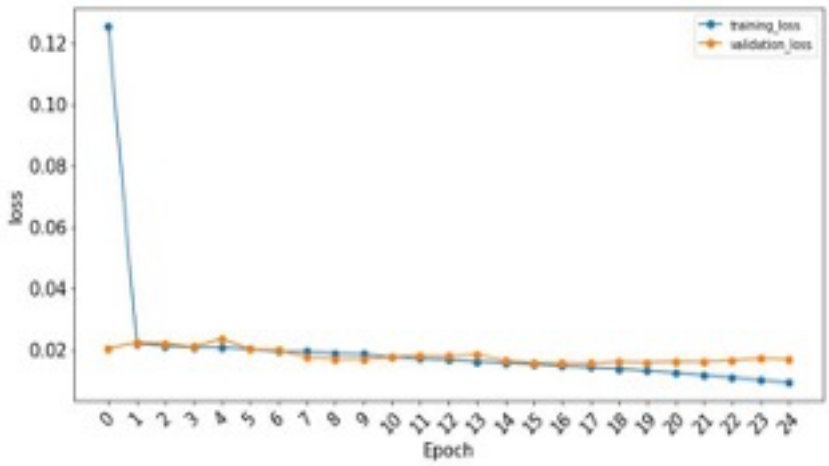

Figure 1: Training and validation loss using architecture 1

```
True Age   : 34.00, 18.00, 32.00, 43.00, 38.00, 28.00, 33.00, 27.00,
Predict Age: 37.73, 29.25, 44.77, 38.95, 35.64, 26.66, 41.34, 34.97,
```

Figure 2: True and predicted age with input images using architecture 1

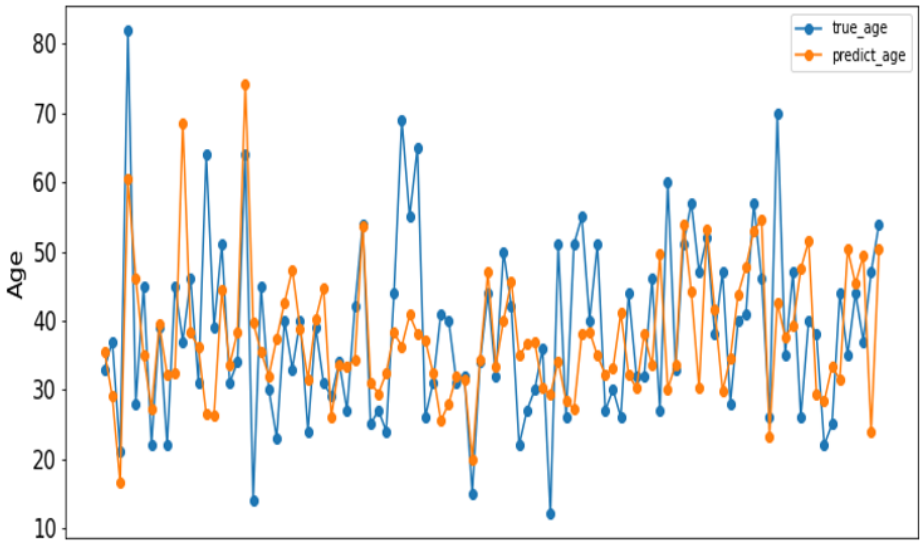

Figure 3: True and predicted age curve obtained using architecture 1

Result Discussion:

For all three architectures, the accuracy of the network is comparatively low. The reasons for this low accuracy in age prediction might be due to the following reasons:

- Only the outliers on age and image size were removed from the data. There might be other outliers such as blurred images, images with more than one face, and others, which could not be removed and might have decreased the efficiency of the model.
- Some of the true labels of age in the image could be incorrect which might have affected the model.
- All input images are grayscale and their size is reduced to 64x64 in order to make the computation easy and fast. This reduction in size and channel on input images may have caused significant loss of information decreasing the efficiency and affecting the accuracy.
- The even number of kernel size and stride size provided more efficiency and accuracy in performance, compared with odd numbers.