# General Guide for Running Standalone Lean Network on AGV Extracted Features

The goal here, essentially, is to train a relatively small network to recognize different age groups. For this, we will be using three sets of features extracted at various stages of a ResNet100 architecture. The original images are passed through a face detector and several layers of ResNet before they are on-boarded to a smaller network.

## List of Datasets Used

- https://github.com/JingchunCheng/All-Age-Faces-Dataset: This contains around 13,300 images of primarily Asian faces. It has only age labels ranging from 3 to 80. Using only this to train or test can lead to a hugely biased network.

- https://susanqq.github.io/UTKFace/: This dataset contains over 20,000 images with age, gender, and ethnicity labels. The corresponding aligned and cropped faces are also provided. It is stored as: [age]_[gender]_[race]_[date&time].jpg .It has a good chunk of Indian data as well.

- http://chalearnlap.cvc.uab.es/dataset/26/description/: This is my standard training dataset. It has around 7,500 images.

- https://data.vision.ee.ethz.ch/cvl/rrothe/imdb-wiki/: By far the most comprehensive dataset with over 500,000 images. Gender and age labels are available. However, this dataset cannot be used for commercial purposes. It can be used for various experiments.

- https://www.kaggle.com/frabbisw/facial-age: This is an excellent dataset that should be added to the training set. Since it has a good number of images for teenagers, it could potentially eliminate the current bias we are facing.

## List of additional experiments conducted

- Age and gender with ResNet50, Inception Resnet V2, ResNet34, and Xception Net
- Age estimation with ResNet 50 frozen layers
- Feature Extraction
- Standalone network trained on features
- Regression analysis

- Additional: - Age and gender training in Caffe with Tal Hasner network (Prediction in terms of age ranges)

## List of python dependencies

- Python 3.7+
- argparse
- cv2 (opencv) 4.3.0
- pathlib
- numpy, pandas, math, random, and scipy
- tables
- Tensorflow GPU 2.3.0
- Nvidia CUDA 10.1
- Keras 2.4.3
- sklearn
- matplotlib
- Dlib (optional)
- Augmentor
- PIL
- better_exceptions
- contextlib
- pickle
- time, datetime

## Instructions to Run the Primary Network

Clone this git repository: [https://github.com/HawkSupertramp/age_gender](https://github.com/HawkSupertramp/age_gender). This is a private repository. You will notice that there are two scripts: trials.py and trial_2_post_facto.py. Use the first one to perform pre-facto training and the other to perform post-fact training. All the saved models are stored in the 'models' directory. The script will automatically attempt to import this library. If it does not find it, it will build the model and train it from scratch. The results directory contains plots and confusion matrices.

If you want to plot a comparison bar chart, you will have to manually move the output matrices (.csv files) to the parent 'results' directory from their respective layer directories. Download the features and extract them into this directory. The names should remain 'add_98', 'stage3_unit15_conv1', and 'stage4_unit1_conv1' in order for the script to find the files. These files have to be downloaded, and are not included in the git repository.

The code is mostly self-explanatory. Here are a few points to note:

- There are two functions that can be used to convert the integer labels into categories. Use ‘convert_to_cat7’ to obtain 7 categories, or ‘convert_to_cat5’ for 5 categories. Remember to change the number of dense layer units to 5 or 7 depending on the number of chosen categories.

- Evaluation is also fairly straightforward. The eval(cats) function will evaluate the categories. Set cats to 0 for 5-category evaluation. Set it to any other number for 7-category evaluation. The resulting confusion matrix will be displayed in the terminal. The outputs will be saved as .csv files in the results directory.

## Running the age and gender recognizer in the alternate experiments folder

This is structured in a non-static method format. There are separate classes defined in various files. It is heavily based on the dataset and implementation provided by: [https://github.com/yu4u/age-gender-estimation](https://github.com/yu4u/age-gender-estimation)

The owner of this repository has the following copyright message:
*“Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so.”*

We are free to modify and redistribute it as we please. The link has most of the instructions to run the code. I have made only slight modifications to train the ResNet from scratch.

### Use trained model for demo

Run the demo script with the default image set to webcam or . You can use ‘--image_dir [IMAGE_DIR]’ option to use images in the specified directory instead. The trained model will be automatically downloaded to the pretrained_models directory.

### Create training data from the IMDB-WIKI dataset

First, download the dataset. The dataset is downloaded and extracted to the data directory by ./download.sh
If this does not work, open the download.sh file in a text editor and manually copy-paste the links into your browser. The dataset download will begin automatically. Filter out noise data and serialize labels into .csv files. Use the following commands to create a trainable database:

*python create_db.py --db imdb*
*usage: create_db.py [-h] [--db DB] [--min_score MIN_SCORE]*

This script cleans-up noisy labels and creates a database for training.

optional arguments:
  -h, --help            show this help message and exit
  --db DB               dataset; wiki or imdb (default: imdb)
  --min_score MIN_SCORE minimum face_score (default: 1.0)
The resulting files with default parameters are included in this repo (meta/imdb.csv and meta/wiki.csv), thus there is no need to run this by yourself.

Use
https://github.com/yu4u/age-gender-estimation/releases/download/v0.5/weights.29-3.76_utk.hdf5 to download pre-trained weights for the model.

This implementation allows you to freely choose the model to rain. You can simply use the following train command to change the model.

```
python train.py model.model_name=ResNet50 model.batch_size=64
```

Currently, ResNet50, ResNet34, ExceptionNet, and InceptionResnetV2 are supported. Note that you will have to specify the input shapes for ExceptionNet and InceptionResnetV2 in the models.py file.

Y4u4's repository asks you to login to wandb to view the results. I do not recommend this. Simply specify the metrics in your training callbacks and plot it using matplotlib. For this, you can use my primary network code.

To simply use the trained model and visualize a few outputs, you can run demo.py. I have modified this slightly.

```
python demo.py

usage: demo.py [-h] [--weight_file WEIGHT_FILE] [--margin MARGIN]
               [--image_dir IMAGE_DIR]

This script detects faces from webcam input, and estimates age and gender for
the detected faces.

optional arguments:
  -h, --help            show this help message and exit
  --weight_file WEIGHT_FILE
                        path to weight file (e.g. weights.28-3.73.hdf5)
                        (default: None)
  --margin MARGIN       margin around detected face for age-gender estimation
                        (default: 0.4)
  --image_dir IMAGE_DIR
                        target image directory; if set, images in image_dir
                        are used instead of webcam (default: None)
```

## Additional Helpful Scripts

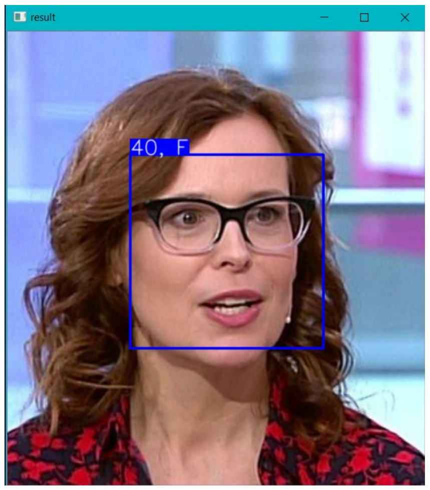
- There is a random_gen.py script in the additional experiments folder. This is a standalone script that I often used to view random samples in the imdb_wiki database. With slight modifications, it can be used to randomly generate 1000 samples from any database. It's fairly simple and can be used for statistical verification of training/test data.

- There is load_data.py script that you can use to load the images and labels, add some noise and augmentation, detect faces, and save a shuffled set of labels + cropped images in pickle files. With slight modifications, this can be used to import and prepare training data for any dataset.

- The im_gen.py is a standard generator skeleton that can be applied to any database. It's a good way to bootstrap any fit_generator functions in the train.py script.

## Results for future comparisons:

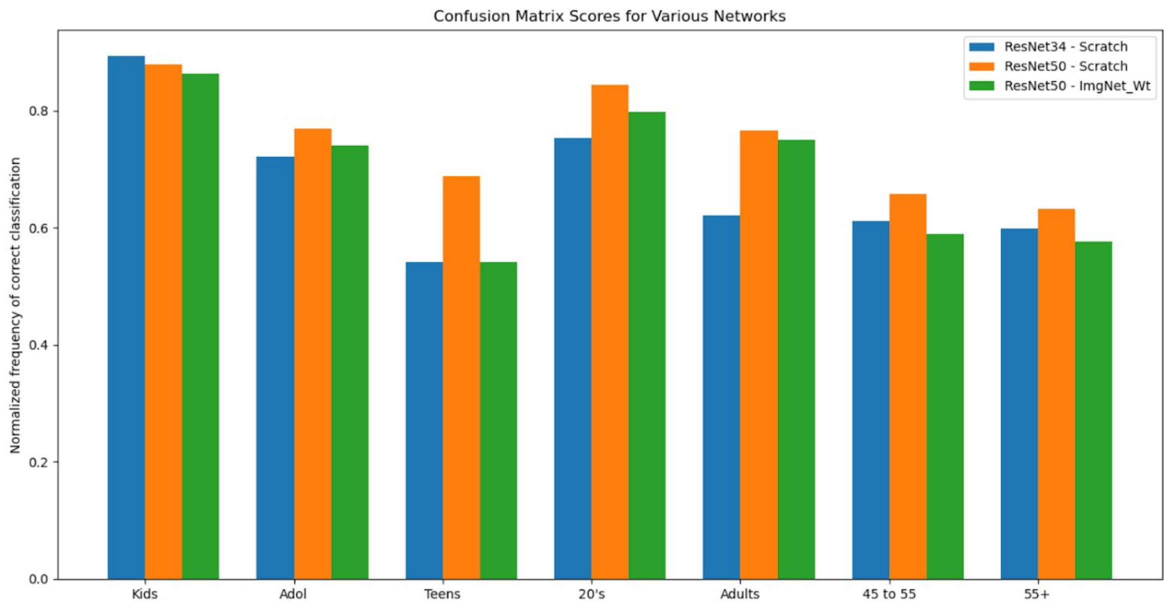
- Dlib pipeline with ResNet50-scratch sample output:



-

The ‘F’ represents ‘female’.

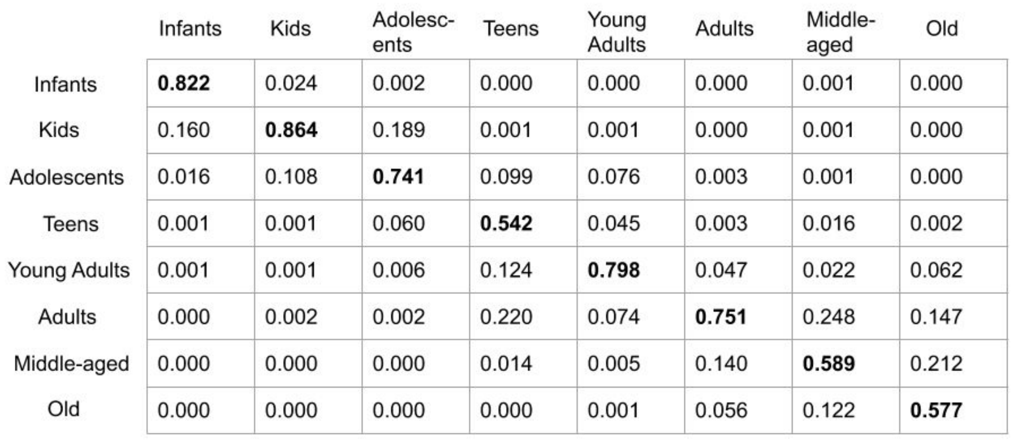
- Old ResNet34, ResNet50, and imported features comparison of accuracies:



- Confusion Matrix results for ResNet50 with Frozen Layers:

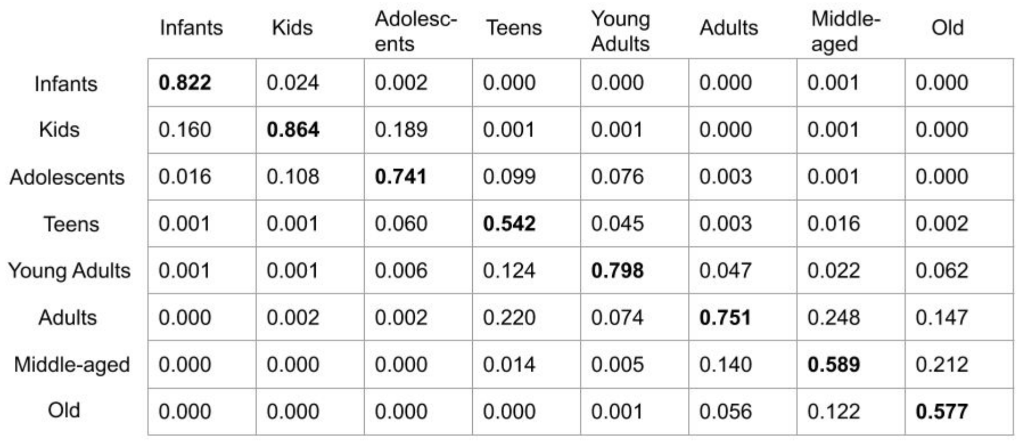
| | Infants | Kids | Adolesc-ents | Teens | Young Adults | Adults | Middle-aged | Old |
|---|---|---|---|---|---|---|---|---|
| Infants | **0.822** | 0.024 | 0.002 | 0.000 | 0.000 | 0.000 | 0.001 | 0.000 |
| Kids | 0.160 | **0.864** | 0.189 | 0.001 | 0.001 | 0.000 | 0.001 | 0.000 |
| Adolescents | 0.016 | 0.108 | **0.741** | 0.099 | 0.076 | 0.003 | 0.001 | 0.000 |
| Teens | 0.001 | 0.001 | 0.060 | **0.542** | 0.045 | 0.003 | 0.016 | 0.002 |
| Young Adults | 0.001 | 0.001 | 0.006 | 0.124 | **0.798** | 0.047 | 0.022 | 0.062 |
| Adults | 0.000 | 0.002 | 0.002 | 0.220 | 0.074 | **0.751** | 0.248 | 0.147 |
| Middle-aged | 0.000 | 0.000 | 0.000 | 0.014 | 0.005 | 0.140 | **0.589** | 0.212 |
| Old | 0.000 | 0.000 | 0.000 | 0.000 | 0.001 | 0.056 | 0.122 | **0.577** |

- Original ResNet50 from Scratch trained on 450,000 images and tested on 36,000 samples:



| | Infants | Kids | Adolesc-ents | Teens | Young Adults | Adults | Middle-aged | Old |
|---|---|---|---|---|---|---|---|---|
| Infants | **0.822** | 0.024 | 0.002 | 0.000 | 0.000 | 0.000 | 0.001 | 0.000 |
| Kids | 0.160 | **0.864** | 0.189 | 0.001 | 0.001 | 0.000 | 0.001 | 0.000 |
| Adolescents | 0.016 | 0.108 | **0.741** | 0.099 | 0.076 | 0.003 | 0.001 | 0.000 |
| Teens | 0.001 | 0.001 | 0.060 | **0.542** | 0.045 | 0.003 | 0.016 | 0.002 |
| Young Adults | 0.001 | 0.001 | 0.006 | 0.124 | **0.798** | 0.047 | 0.022 | 0.062 |
| Adults | 0.000 | 0.002 | 0.002 | 0.220 | 0.074 | **0.751** | 0.248 | 0.147 |
| Middle-aged | 0.000 | 0.000 | 0.000 | 0.014 | 0.005 | 0.140 | **0.589** | 0.212 |
| Old | 0.000 | 0.000 | 0.000 | 0.000 | 0.001 | 0.056 | 0.122 | **0.577** |

- Comparison of various networks with augmented features/feature extraction:



| | Infants | Kids | Adolesc-ents | Teens | Young Adults | Adults | Middle-aged | Old |
|---|---|---|---|---|---|---|---|---|
| Infants | **0.822** | 0.024 | 0.002 | 0.000 | 0.000 | 0.000 | 0.001 | 0.000 |
| Kids | 0.160 | **0.864** | 0.189 | 0.001 | 0.001 | 0.000 | 0.001 | 0.000 |
| Adolescents | 0.016 | 0.108 | **0.741** | 0.099 | 0.076 | 0.003 | 0.001 | 0.000 |
| Teens | 0.001 | 0.001 | 0.060 | **0.542** | 0.045 | 0.003 | 0.016 | 0.002 |
| Young Adults | 0.001 | 0.001 | 0.006 | 0.124 | **0.798** | 0.047 | 0.022 | 0.062 |
| Adults | 0.000 | 0.002 | 0.002 | 0.220 | 0.074 | **0.751** | 0.248 | 0.147 |
| Middle-aged | 0.000 | 0.000 | 0.000 | 0.014 | 0.005 | 0.140 | **0.589** | 0.212 |
| Old | 0.000 | 0.000 | 0.000 | 0.000 | 0.001 | 0.056 | 0.122 | **0.577** |

**Links to Reference Material:**

- CVPR 2015 - Hassner - https://talhassner.github.io/home/publication/2015_CVPR

- ICCIDS 2018 - https://www.researchgate.net/publication/325664314_Gender_Recognition_Through_Face_Using_Deep_Learning

- ETH Zurich (DeX: VGG16 architecture) - https://www.researchgate.net/publication/283356929_DEX_Deep_EXpectation_of_Apparent_Age_from_a_Single_Image

- OpenCV implementation - https://www.pyimagesearch.com/2020/04/13/opencv-age-detection-with-deep-learning/

- Another OpenCV implementation - https://towardsdatascience.com/predict-age-and-gender-using-convolutional-neural-network-and-opencv-fd90390e3ce6

- Wide Residual Networks (2017) - https://arxiv.org/pdf/1605.07146.pdf

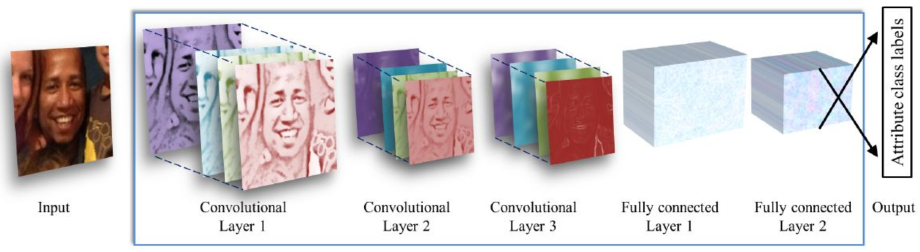
- Rank-consistent Ordinal Regression for Neural Networks (2019) - https://arxiv.org/pdf/1901.07884v5.pdf

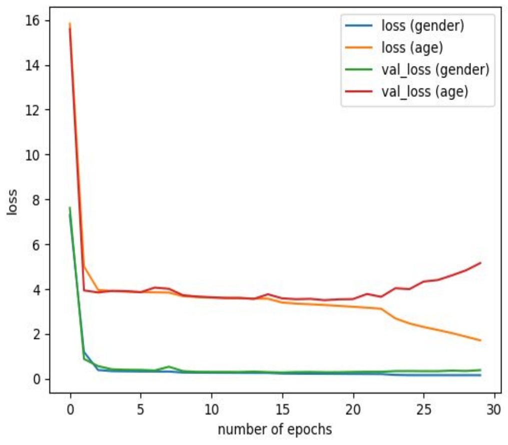
# Appendix - Collection of Past Reports and References

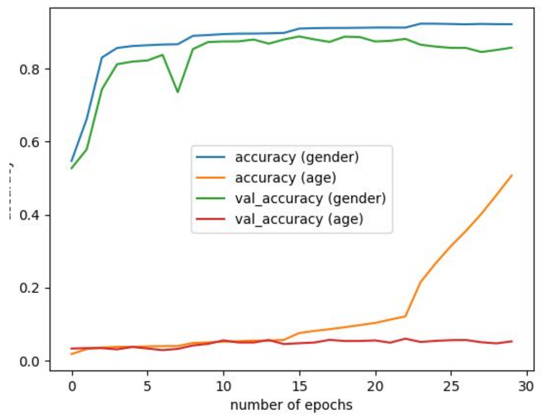
## Report 1:

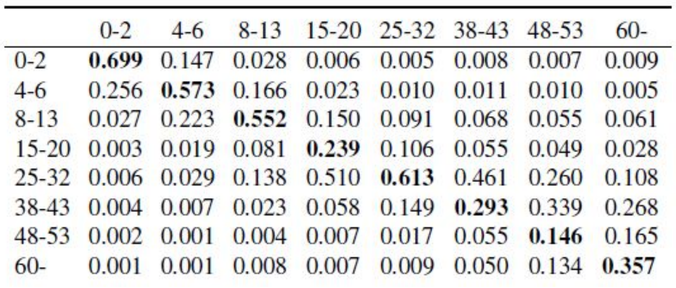
### Levi-Hassner Network Architecture (CVPR - 2015):

https://talhassner.github.io/home/publication/2015_CVPR









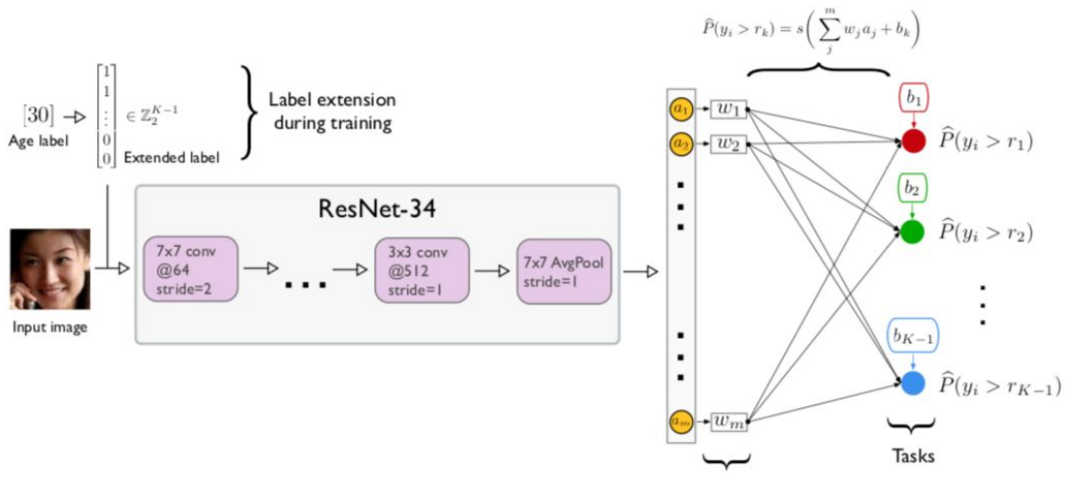
| | 0-2 | 4-6 | 8-13 | 15-20 | 25-32 | 38-43 | 48-53 | 60- |
|---|---|---|---|---|---|---|---|---|
| 0-2 | **0.699** | 0.147 | 0.028 | 0.006 | 0.005 | 0.008 | 0.007 | 0.009 |
| 4-6 | 0.256 | **0.573** | 0.166 | 0.023 | 0.010 | 0.011 | 0.010 | 0.005 |
| 8-13 | 0.027 | 0.223 | **0.552** | 0.150 | 0.091 | 0.068 | 0.055 | 0.061 |
| 15-20 | 0.003 | 0.019 | 0.081 | **0.239** | 0.106 | 0.055 | 0.049 | 0.028 |
| 25-32 | 0.006 | 0.029 | 0.138 | 0.510 | **0.613** | 0.461 | 0.260 | 0.108 |
| 38-43 | 0.004 | 0.007 | 0.023 | 0.058 | 0.149 | **0.293** | 0.339 | 0.268 |
| 48-53 | 0.002 | 0.001 | 0.004 | 0.007 | 0.017 | 0.055 | **0.146** | 0.165 |
| 60- | 0.001 | 0.001 | 0.008 | 0.007 | 0.009 | 0.050 | 0.134 | **0.357** |

Age Ranges:

- Infants (0 to 2)
- Kids (4 to 6)
- Adolescents (8 to 13)
- Teens (14 to 20)
- Young Adults (25 to 32)
- Adults (38 to 43)
- Middle-aged (48 to 53)
- Old (60+)

## ResNet Implementation:

Architecture:



**Datasets:**

- CACD Dataset : https://bcsiriuschen.github.io/CARC/
- UTK Face
- MORPH2
- AFAD

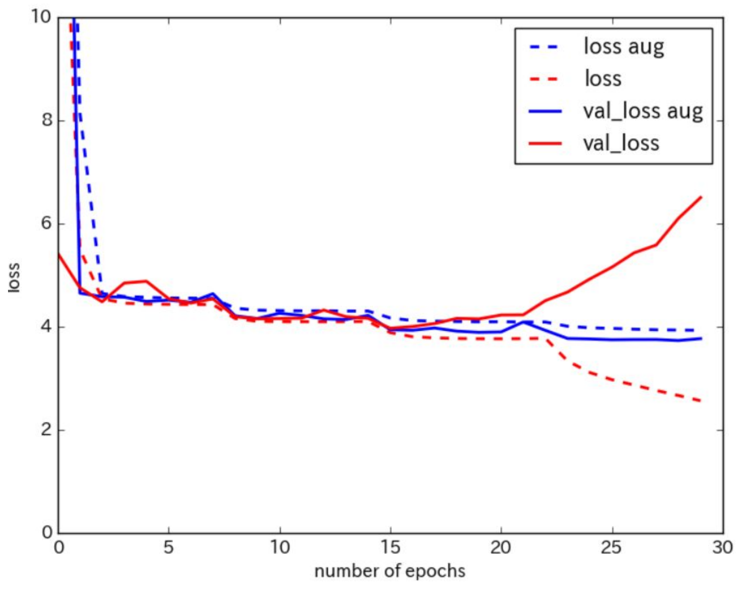
```
Time elapsed: 519.53 min
MAE/RMSE: | Train: 0.73/2.12 | Valid: 5.63/8.02 | Test: 5.72/8.07
Total Training Time: 520.49 min
MAE/RMSE: | Best Train: 1.93/2.86 | Best Valid: 5.27/7.57 | Best Test:
5.37/7.70
```

**Confusion Matrix:**

| | 0 - 2 | 4 - 6 | 8 - 14 | 15 - 20 | 22 - 32 | 38 - 43 | 48 - 53 | 55+ |
|---|---|---|---|---|---|---|---|---|
| 0 - 2 | **0.732** | 0.052 | 0.002 | 0.002 | 0.001 | 0.001 | 0.001 | 0.001 |
| 4 - 6 | 0.266 | **0.689** | 0.148 | 0.029 | 0.001 | 0.002 | 0.001 | 0.002 |
| 8 - 14 | 0.002 | 0.246 | **0.546** | 0.288 | 0.062 | 0.004 | 0.008 | 0.009 |
| 15 - 20 | 0.001 | 0.012 | 0.293 | **0.322** | 0.210 | 0.001 | 0.009 | 0.016 |
| 22 - 32 | 0.001 | 0.001 | 0.009 | 0.322 | **0.620** | 0.212 | 0.028 | 0.119 |
| 38 - 43 | 0.000 | 0.000 | 0.002 | 0.034 | 0.100 | **0.342** | 0.252 | 0.224 |
| 48 - 53 | 0.000 | 0.000 | 0.001 | 0.002 | 0.005 | 0.121 | **0.390** | 0.321 |
| 55+ | 0.000 | 0.000 | 0.001 | 0.001 | 0.001 | 0.001 | 0.311 | **0.398** |

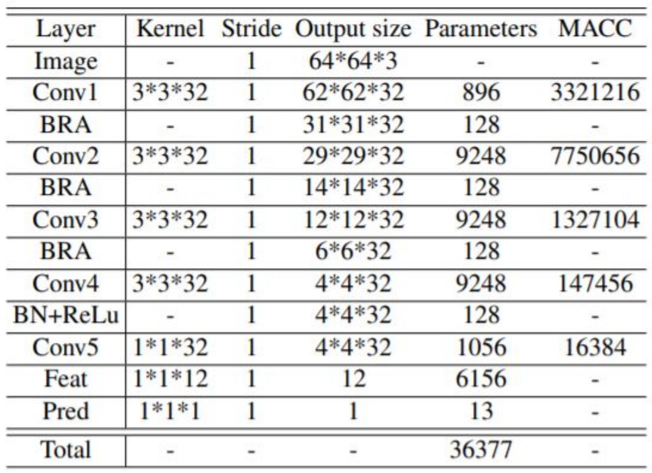
Note that this confusion matrix was the result of faulty noise addition. This noise addition was later corrected. You can find the updated noise addition and augmentation techniques in ./alternate_experiments/age_gender/age_estimation/view_augmented.py. Simply generate a random set of images and store them in the /RDM_Images_IMDB folder in this directory.

The view_augmented.py is a custom script not included in y4u4 repository. It is specifically used to experiment with the dlib face detector and various augmentation techniques.



**C3AE Implementation:**

Architecture:



| Layer | Kernel | Stride | Output size | Parameters | MACC |
|---|---|---|---|---|---|
| Image | - | 1 | 64*64*3 | - | - |
| Conv1 | 3*3*32 | 1 | 62*62*32 | 896 | 3321216 |
| BRA | - | 1 | 31*31*32 | 128 | - |
| Conv2 | 3*3*32 | 1 | 29*29*32 | 9248 | 7750656 |
| BRA | - | 1 | 14*14*32 | 128 | - |
| Conv3 | 3*3*32 | 1 | 12*12*32 | 9248 | 1327104 |
| BRA | - | 1 | 6*6*32 | 128 | - |
| Conv4 | 3*3*32 | 1 | 4*4*32 | 9248 | 147456 |
| BN+ReLu | - | 1 | 4*4*32 | 128 | - |
| Conv5 | 1*1*32 | 1 | 4*4*32 | 1056 | 16384 |
| Feat | 1*1*12 | 1 | 12 | 6156 | - |
| Pred | 1*1*1 | 1 | 1 | 13 | - |
| Total | - | - | - | 36377 | - |

## Report 2:

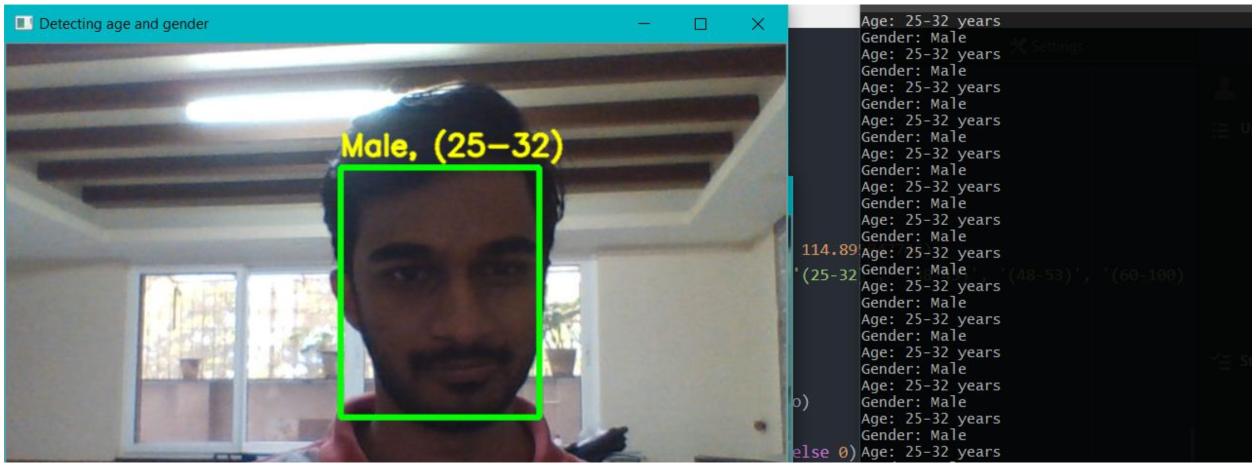
### Choosing a linear classifier for face detection

- Haar cascades will be very fast and capable of running in real-time on embedded devices — the problem is that they are less accurate and highly prone to false-positive detections. I initially used this, but since you will not be running on an embedded device, I thought it would be better to take a deep learning-based compute-intensive approach for detection.

- HOG + Linear SVM models are more accurate than Haar cascades but are slower. They also aren't as tolerant with occlusion (i.e., not all of the face visible) or viewpoint changes (i.e., different views of the face). **In my implementation, I have not used this since I ran into problems with occlusion.**

- Deep learning-based face detectors are the most robust and will give the best accuracy, but require even more computational resources than both Haar cascades and HOG + Linear SVMs.

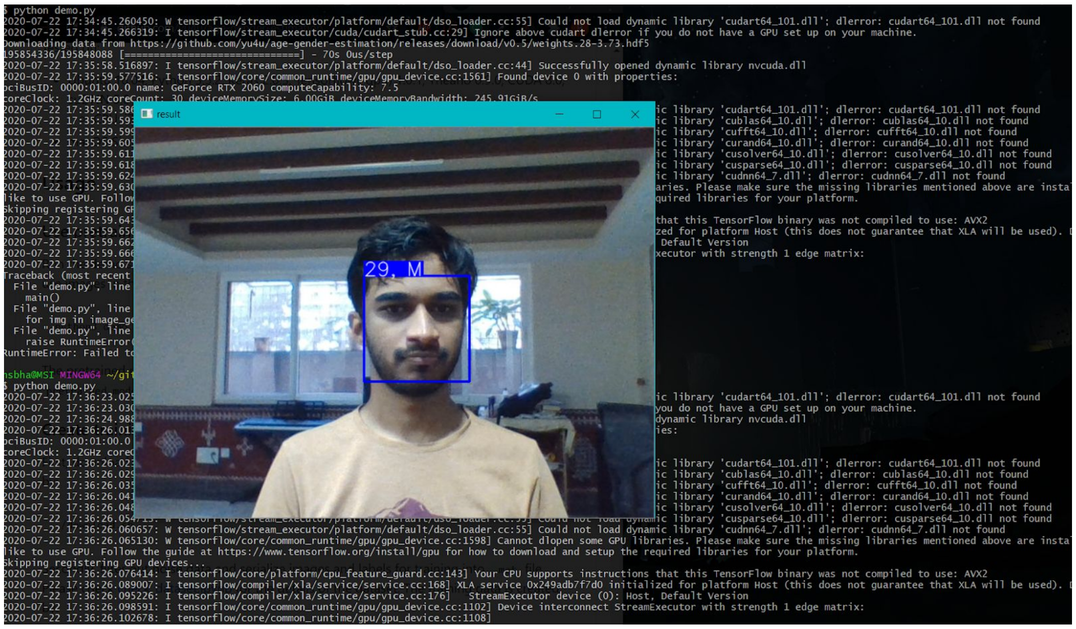
### Sample Output (Range - Caffe + OpenCV Implementation):



```
114.89
'(25-32                                  '(48-53)', '(60-100)
o)
else 0)
```

```
Age: 25-32 years
Gender: Male
Age: 25-32 years
Gender: Male
Age: 25-32 years
Gender: Male
Age: 25-32 years
Gender: Male
Age: 25-32 years
Gender: Male
Age: 25-32 years
Gender: Male
Age: 25-32 years
Gender: Male
Age: 25-32 years
Gender: Male
Age: 25-32 years
Gender: Male
Age: 25-32 years
Gender: Male
Age: 25-32 years
Gender: Male
Age: 25-32 years
Gender: Male
Age: 25-32 years
Gender: Male
Age: 25-32 years
```

Sample Output (Specific Value - Keras Implementation):



```
$ python demo.py
2020-07-22 17:34:45.260450: W tensorflow/stream_executor/platform/default/dso_loader.cc:55] Could not load dynamic library 'cudart64_101.dll'; dlerror: cudart64_101.dll not found
2020-07-22 17:34:45.266319: I tensorflow/stream_executor/cuda/cudart_stub.cc:29] Ignore above cudart dlerror if you do not have a GPU set up on your machine.
Downloading data from https://github.com/yu4u/age-gender-estimation/releases/download/v0.5/weights.28-3.73.hdf5
195854336/195848088 [==============================] - 70s 0us/step
2020-07-22 17:35:58.516897: I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic library nvcuda.dll
2020-07-22 17:35:59.577516: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1561] Found device 0 with properties:
pciBusID: 0000:01:00.0 name: GeForce RTX 2060 computeCapability: 7.5
coreClock: 1.2GHz coreCount: 30 deviceMemorySize: 6.00GiB deviceMemoryBandwidth: 245.91GiB/s
2020-07-22 17:35:59.586                                            ic library 'cudart64_101.dll'; dlerror: cudart64_101.dll not found
2020-07-22 17:35:59.593                                            ic library 'cublas64_10.dll'; dlerror: cublas64_10.dll not found
2020-07-22 17:35:59.599                                            ic library 'cufft64_10.dll'; dlerror: cufft64_10.dll not found
2020-07-22 17:35:59.605                                            ic library 'curand64_10.dll'; dlerror: curand64_10.dll not found
2020-07-22 17:35:59.611                                            ic library 'cusolver64_10.dll'; dlerror: cusolver64_10.dll not found
2020-07-22 17:35:59.618                                            ic library 'cusparse64_10.dll'; dlerror: cusparse64_10.dll not found
2020-07-22 17:35:59.624                                            ic library 'cudnn64_7.dll'; dlerror: cudnn64_7.dll not found
2020-07-22 17:35:59.630                                            aries. Please make sure the missing libraries mentioned above are insta
like to use GPU. Follow                                            quired libraries for your platform.
Skipping registering GP
2020-07-22 17:35:59.643                                            that this TensorFlow binary was not compiled to use: AVX2
2020-07-22 17:35:59.656                                            zed for platform Host (this does not guarantee that XLA will be used). 
2020-07-22 17:35:59.662                                             Default Version
2020-07-22 17:35:59.666                                            xecutor with strength 1 edge matrix:
2020-07-22 17:35:59.671
Traceback (most recent
  File "demo.py", line
    main()
  File "demo.py", line
    for img in image_ge
  File "demo.py", line
    raise RuntimeError(
RuntimeError: Failed to

nsbha@MSI MINGW64 ~/git
$ python demo.py
2020-07-22 17:36:23.025                                            ic library 'cudart64_101.dll'; dlerror: cudart64_101.dll not found
2020-07-22 17:36:23.030                                            you do not have a GPU set up on your machine.
2020-07-22 17:36:24.988                                            dynamic library nvcuda.dll
2020-07-22 17:36:26.013                                            ies:
pciBusID: 0000:01:00.0
coreClock: 1.2GHz core
2020-07-22 17:36:26.023                                            ic library 'cudart64_101.dll'; dlerror: cudart64_101.dll not found
2020-07-22 17:36:26.029                                            ic library 'cublas64_10.dll'; dlerror: cublas64_10.dll not found
2020-07-22 17:36:26.035                                            ic library 'cufft64_10.dll'; dlerror: cufft64_10.dll not found
2020-07-22 17:36:26.041                                            ic library 'curand64_10.dll'; dlerror: curand64_10.dll not found
2020-07-22 17:36:26.048                                            ic library 'cusolver64_10.dll'; dlerror: cusolver64_10.dll not found
2020-07-22 17:36:26.054                                            ic library 'cusparse64_10.dll'; dlerror: cusparse64_10.dll not found
2020-07-22 17:36:26.060657: W tensorflow/stream_executor/platform/default/dso_loader.cc:55] Could not load dynamic library 'cudnn64_7.dll'; dlerror: cudnn64_7.dll not found
2020-07-22 17:36:26.065130: W tensorflow/core/common_runtime/gpu/gpu_device.cc:1598] Cannot dlopen some GPU libraries. Please make sure the missing libraries mentioned above are insta
like to use GPU. Follow the guide at https://www.tensorflow.org/install/gpu for how to download and setup the required libraries for your platform.
Skipping registering GPU devices...
2020-07-22 17:36:26.076414: I tensorflow/core/platform/cpu_feature_guard.cc:143] Your CPU supports instructions that this TensorFlow binary was not compiled to use: AVX2
2020-07-22 17:36:26.089007: I tensorflow/compiler/xla/service/service.cc:168] XLA service 0x249adb7f7d0 initialized for platform Host (this does not guarantee that XLA will be used). 
2020-07-22 17:36:26.095226: I tensorflow/compiler/xla/service/service.cc:176]   StreamExecutor device (0): Host, Default Version
2020-07-22 17:36:26.098591: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1102] Device interconnect StreamExecutor with strength 1 edge matrix:
2020-07-22 17:36:26.102678: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1108]
```

**Normal Lighting (mis-classified age as 29 - lower error)**
**Inference time: 0.023 seconds**