NANYANG
TECHNOLOGICAL
UNIVERSITY
SINGAPORE

CZ4042 Neural Networks and Deep Learning

Topic D. Gender Classification

Wee Chang Han U2022364G

Saravanabavan Varsha U2020628H

Banerjee Tanya U2023315L

Table of Contents

1. Introduction

1.1 Background and Motivation

The growing importance of identifying a person's age and gender is driven by a wide range of applications in our modern society, including but not limited to security systems and healthcare. In the ever-evolving field of classification techniques, machine learning models like Convolutional Neural Networks (CNN) have gained prominence, underscoring the significance of a person's gender and age as key factors in a multivariate equation for identification in our increasingly modernised world.

1.2 Project Scope

In this paper, we aim to comprehensively investigate how traditional CNN and state-of-the-art image classification models, specifically the ResNet50 and EfficientNet, perform in the classification of gender and age of faces. Furthermore, we will explore age and gender recognition simultaneously through EfficientNet, leveraging gender-specific age characteristics and age-specific gender characteristics inherent in images for efficient classification. We will present a comparative analysis of our results across the various models, showcasing the evolution of their efficacy and efficiency over time. Additionally, we will identify and suggest potential improvements to close the research gap in the field of age and gender classification to contribute to the field's ongoing development.

2. Related Work

2.1 Standard Convolutional Neural Network (CNN)

The motivation and work for CNN's latent potential in age-gender classification was first established in 2015 by Gil Levi and Tal Hassner [1], where a shallow Convolutional Architecture for Fast Feature Embedding (Caffe) deep learning framework was used to train neural network models that focused on Internet images instead of constrained images taken in lab settings. This lent credibility to CNN's effectiveness in producing competitive results on a challenging dataset. While it is important to consider the limitations and potential biases associated with large-scale internet-collected datasets, this paper provided researchers with invaluable data and possible applications in real-world scenarios.

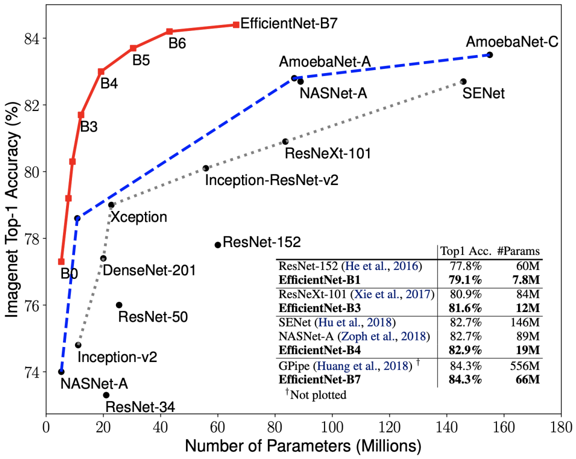
2.2 ResNet-50

Introduced by Kaiming He, Xiangyu Zhang et al [3], ResNet-50 is a 50-layered variation of a convolutional neural network architecture that belongs to the family of Residual Networks (ResNets). The innovation of ResNet architecture with its use of residual blocks to address the vanishing gradient problem in deep neural networks earned its authors 1st places on the tasks of ImageNet detection, ImageNet localization, COCO detection, and COCO segmentation [3]. Henceforth, pre-trained ResNet-50 models were often fine-tuned and used by researchers and practitioners for specific tasks due to their excellent generalisation capabilities.

2.3 EfficientNet

Recently introduced by a team of researchers at Google AI, EfficientNet has become a go-to architecture for various computer vision tasks. EfficientNet is a powerful CNN architecture that is designed to optimise the network's depth, width, and resolution simultaneously. It uses a combination of neural architecture search (NAS) and model scaling to achieve both higher accuracy and efficiency on 5 out of 8 widely used datasets while reducing parameters by up to 21x than existing CNNs [4]. The model's lightweight and robust design brings great potential for real-world industrial applications with various hardware constraints.

Figure 2.3.1 shows the performance of the EfficientNet family of models compared to other network architectures [4]. There are currently 8 different models that are in order of increasing number of trainable parameters, namely B0-B7. EfficientNet B0 is the baseline network while B1-B7 are scaled up variants. The biggest EfficientNet model EfficientNet B7 obtained state-of-the-art performance on the ImageNet and the CIFAR-100 datasets.



	Top1 Acc.	#Params
ResNet-152 (He et al., 2016)	77.8%	60M
EfficientNet-B1	**79.1%**	**7.8M**
ResNeXt-101 (Xie et al., 2017)	80.9%	84M
EfficientNet-B3	**81.6%**	**12M**
SENet (Hu et al., 2018)	82.7%	146M
NASNet-A (Zoph et al., 2018)	82.7%	89M
EfficientNet-B4	**82.9%**	**19M**
GPipe (Huang et al., 2018) †	84.3%	556M
EfficientNet-B7	**84.3%**	**66M**

†Not plotted

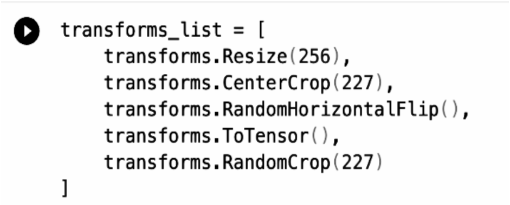
Figure 2.3.1 Accuracies of Network Models vs Number of Parameters [4]

3. Data Handling

In this section, we will explain the preprocessing done on the dataset that will be used in our models.

3.1 Adience Dataset

The Adience Dataset contains a total of 26580 in-the-wild images having variations in appearance, noise, pose and lighting. The images we used have been cropped and aligned, with each image assigned to one of eight age groups (0-2, 4-6, 8-13, 15-20, 25-32, 38-43, 48-53 or >60) and one of two gender labels (M or F).



```
transforms_list = [
    transforms.Resize(256),
    transforms.CenterCrop(227),
    transforms.RandomHorizontalFlip(),
    transforms.ToTensor(),
    transforms.RandomCrop(227)
]
```

Figure 3.1.1 Adience Dataset Preparation & Augmentation

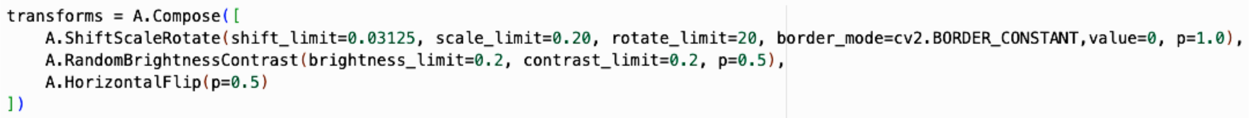
As shown in Figure 3.1.1, the following methods were used for data preparation and augmentation:

- Resized images to 256 x 256 pixels
- Cropped images to retain only 227 x 227 pixels from the centre
- Randomly performed horizontal flip on images
- Cropped images to retain 227 x 227 pixels from a random area

For the Levi-Hassner CNN, our training involved using the Adience dataset exclusively. The same dataset was then used for transfer learning on the ResNet-50 model to fine-tune the pretrained model that was initialised with ImageNet weights.

3.2 IMDb-Wiki Dataset

The IMDb-Wiki Dataset is the largest publicly available dataset of face images with gender and age labels for training purposes. There are 460,723 images in the IMBb dataset and 62,328 images in the Wiki dataset. Due to computational constraints, we limited our training data to a set of 60,000 images by employing a seeded random selection process. In our experiment, we have used the IMDb-Wiki dataset for transfer learning purposes to pretrain our EfficientNet Model.



```
transforms = A.Compose([
    A.ShiftScaleRotate(shift_limit=0.03125, scale_limit=0.20, rotate_limit=20, border_mode=cv2.BORDER_CONSTANT,value=0, p=1.0),
    A.RandomBrightnessContrast(brightness_limit=0.2, contrast_limit=0.2, p=0.5),
    A.HorizontalFlip(p=0.5)
])
```

Figure 3.2.1 IMDb-Wiki Dataset Preparation & Augmentation

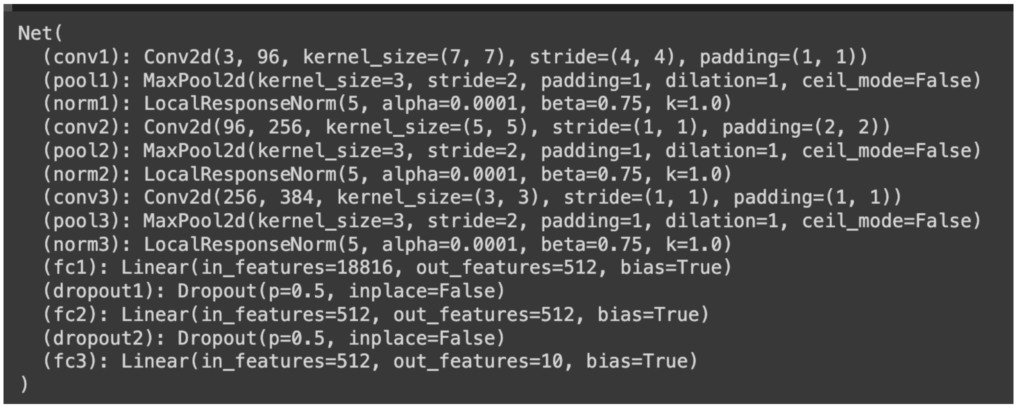
Shown in Figure 3.2.1, a series of image augmentations using the Albumentations library was introduced to create more variations in the dataset.

- Resized images to 224 x 224 pixels to match input shape for model
- ShiftScaleRotate: Shifting, scaling and rotating the image up to a certain factor
- RandomBrightnessContrast: Random brightness and contrast adjustments
- HorizontalFlip: Flips the image horizontally with a 50% chance

4. Methodology & Results

4.1 Levi-Hassner CNN

We implemented the traditional CNN proposed by G. Levi and T. Hassner in [1] and trained and tested on the Adience dataset.



```
Net(
  (conv1): Conv2d(3, 96, kernel_size=(7, 7), stride=(4, 4), padding=(1, 1))
  (pool1): MaxPool2d(kernel_size=3, stride=2, padding=1, dilation=1, ceil_mode=False)
  (norm1): LocalResponseNorm(5, alpha=0.0001, beta=0.75, k=1.0)
  (conv2): Conv2d(96, 256, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
  (pool2): MaxPool2d(kernel_size=3, stride=2, padding=1, dilation=1, ceil_mode=False)
  (norm2): LocalResponseNorm(5, alpha=0.0001, beta=0.75, k=1.0)
  (conv3): Conv2d(256, 384, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (pool3): MaxPool2d(kernel_size=3, stride=2, padding=1, dilation=1, ceil_mode=False)
  (norm3): LocalResponseNorm(5, alpha=0.0001, beta=0.75, k=1.0)
  (fc1): Linear(in_features=18816, out_features=512, bias=True)
  (dropout1): Dropout(p=0.5, inplace=False)
  (fc2): Linear(in_features=512, out_features=512, bias=True)
  (dropout2): Dropout(p=0.5, inplace=False)
  (fc3): Linear(in_features=512, out_features=10, bias=True)
)
```

Figure 4.1.2 Model architecture for Levi Hassner CNN

We developed two distinct models for age and gender classification. For each of these classification problems, we used the model that demonstrated the best performance based on the smallest validation error. We used a batch size of 50 and an initial learning rate of 0.0001 as proposed in the paper, and the LogSoftmax function to calculate probabilities along with Negative Log-Likelihood Loss (NLLL) function. The model then classifies images to one of eight age classes and one of two gender classes.

Figure 4.1.3 shows the table of results we obtained after a 5 epoch training cycle. This will be henceforth used as the basis for our comparison.

Levi-Hassner CNN (5 Epoch)	Training time	Accuracy
Model for age	6914 sec	42.6%
Model for gender	3958 sec	68.0%

Figure 4.1.3 Table of Results for Levi-Hassner CNN

4.2 ResNet-50

Employing a ResNet-50 model, we used the same hyperparameters described in Levi Hassner CNN for the ResNet-50 model. The following were parameterized: a batch size of 50, an initial learning rate of 0.0001, the LogSoftmax function along with NLLLoss and 5 epochs of training.

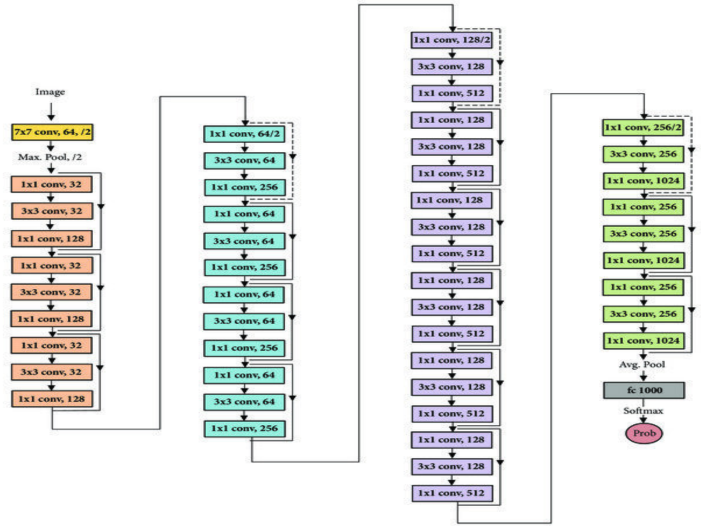


Figure 4.2.1 ResNet-50 model architecture [4]

Similar to the Levi-Hassner CNN, we trained two separate models to predict age and gender independently. We loaded the ResNet50 model with ImageNet weights from torchvision.models and removed the last fully connected layer. Since the last layer is a dense layer with the number of units that equals the number of classes in the original task (i.e 1000 classes for ImageNet), we have replaced it with a new task specific layer that matches the numbers of classes in Adience. Therefore, for each of these two models, we added a custom layer having 2048 x num_classes weights (where num_classes is 8 for age and 2 for gender). This was followed by taking the LogSoftmax of the outputs to get prediction probabilities. We can see that though the ResNet-50 model trains only a single layer, it requires a longer training time than the Levi Hassner model.

ResNet-50	Training time	Accuracy
Model for age	8464 sec	52.36%
Model for gender	5441 sec	76.30%

Figure 4.2.1 Table of Results for ResNet-50

4.3 EfficientNet

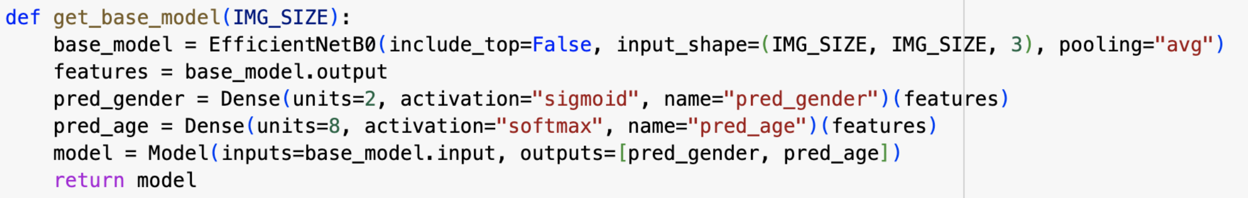
4.3.1 Transfer Learning with EfficientNet

We explore Transfer Learning, a machine learning technique where a pretrained model is adapted or fine tuned for a new related task. It leverages on knowledge gained from the original task to accelerate learning on the new task, leading to improved performance especially in cases where the original dataset is limited. We opted for EfficientNetB0 due to its optimal combination of low parameter size and high performance. To summarise the steps for instantiating an EfficientNetB0 model:

1. Instantiate EfficientNetB0 model with ImageNet weights
2. Train the model on a large dataset (IMDb-Wiki Dataset)
3. Save the model with the highest validation accuracy during training
4. Load the model, remove the last task-specific output layer and freeze all layers so that weights will not be updated during training.
5. Add a new Softmax layer and train the model on the Adience Dataset
6. Experiment with various hyperparameters and architectural layers
7. Obtain EfficientNetB0 with optimal performance and accuracy

4.3.2 Pre-training EfficientNetB0 on IMDb-Wiki Dataset

The IMDB-Wiki Dataset is preprocessed by filtering out noise data and serialising its labels in .csv files for training. Since the age labels for this dataset range from 1 to 102, we manually categorised the ages into 8 categories following the Adience dataset.



```
def get_base_model(IMG_SIZE):
    base_model = EfficientNetB0(include_top=False, input_shape=(IMG_SIZE, IMG_SIZE, 3), pooling="avg")
    features = base_model.output
    pred_gender = Dense(units=2, activation="sigmoid", name="pred_gender")(features)
    pred_age = Dense(units=8, activation="softmax", name="pred_age")(features)
    model = Model(inputs=base_model.input, outputs=[pred_gender, pred_age])
    return model
```

Figure 4.3.1 EfficientNetB0 model for Pre-training

Figure 4.3.1 shows our base EfficientNetB0 model that will be pre-trained. After training, we obtained validation accuracy of 96.44% for gender and 61.93% for age, which was saved to be further trained and fine tuned on the Adience dataset.

4.3.3 Fine Tuning EfficientNetB0

We loaded our pre-trained base model to be trained on the Adience dataset. To improve the model's performance, we experimented with various hyperparameters and techniques.

Batch Size

We can observe that using 64 as our batch size provides the best accuracy for both gender and age prediction. Using 128 as batch size unfortunately gave us a resource exhaustion error, which may be attributed to GPU cluster's memory exhaustion.

Batch Size	32	64	128
Gender Val Accuracy	97.19%	**97.42%**	Error
Age Val Accuracy	61.16%	**66.52%**	Error

Table 4.3.1 Validation Accuracies across different Batch Sizes

Optimizer

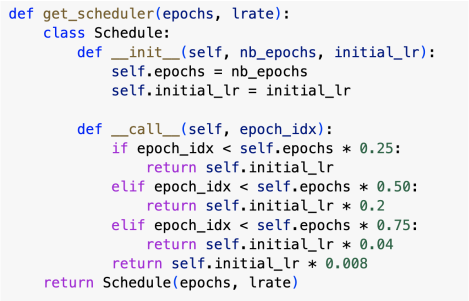
We can observe that using Adam as our optimizer results in the best accuracy for both gender and age prediction

Optimizer	Adam	SGD
Gender Val Accuracy	**97.55%**	92.57%
Age Val Accuracy	**67.14%**	57.57%

Table 4.3.2 Validation Accuracies across different Optimizers

Learning Rate & Learning Rate Scheduler

We have experimented with learning rates of 0.001 and 0.01. Additionally, we have included a learning rate scheduler as seen in Figure 4.3.2, which dynamically adjusts the learning rate during training based on the number of epochs. We observe that implementing a learning rate scheduler provides the best accuracy for both gender and age over fixed learning rates.



```
def get_scheduler(epochs, lrate):
    class Schedule:
        def __init__(self, nb_epochs, initial_lr):
            self.epochs = nb_epochs
            self.initial_lr = initial_lr

        def __call__(self, epoch_idx):
            if epoch_idx < self.epochs * 0.25:
                return self.initial_lr
            elif epoch_idx < self.epochs * 0.50:
                return self.initial_lr * 0.2
            elif epoch_idx < self.epochs * 0.75:
                return self.initial_lr * 0.04
            return self.initial_lr * 0.008
    return Schedule(epochs, lrate)
```

Figure 4.3.2 Learning rate scheduler function

	With LR Scheduler	Without LR Scheduler (LR=0.001)	Without LR Scheduler (LR=0.01)
Gender Val Accuracy	**97.55%**	96.66%	93.27%
Age Val Accuracy	**67.14%**	65.34%	42.98%

Table 4.3.3 Validation Accuracies across different Learning Rates

Activation for Gender Output Layer

We have two output layers for gender and age prediction in our EfficientNet model. Softmax activation is used for predicting age as it is commonly used for multi-class prediction problems (in our case, 8 age categories). Sigmoid activation is typically used for binary classification problems and can model the probability distribution of the output values. We experimented with both Softmax and Sigmoid activation for the gender output layer and we can observe that Sigmoid activation results in a higher gender validation accuracy but a lower age validation accuracy. However, we decided to proceed with Sigmoid activation for gender as it is more appropriate to ensure that the gender predictions are close to 0 or 1.

Activation for Gender	Softmax	Sigmoid
Gender Val Accuracy	97.25%	**97.55%**
Age Val Accuracy	**67.74%**	67.14%

Table 4.3.4 Validation Accuracies across different Activation Layers

4.3.4 EfficientNetB0 Architectural Layers

Dropout mitigates overfitting and encourages robust feature learning, Global Average Pooling reduces model complexity and memory usage, and Batch Normalization stabilises training and accelerates convergence.

Activation for Gender	With Layers	Without Layers
Gender Val Accuracy	94.14%	**97.55%**
Age Val Accuracy	52.23%	**67.14%**

Table 4.3.5 Validation accuracy (With vs Without Additional Layers)

However, we observe that the accuracy drops after implementing these techniques. This may be due to over-regularization where excessive use of dropout layers may lead to underfitting. Furthermore, EfficientNetB0's architecture is already well-optimised and introducing additional complexity with these layers might prove redundant and necessary.

Using the optimal parameters we have experimented on in the previous section, we have found that the best accuracy obtainable is 97.55% for age and 67.14% for gender.

4.3.5 Real Time Prediction using EfficientNet

To highlight the performance of EfficientNetB0, we have developed a program for real time facial detection and prediction of gender and age using the optimal EffiicentNetB0 model.We first instantiate EfficientNetB0 and load weights from the fine tuned model. Next, we access the webcam, capture video frames and use dlib's frontal face detector to capture any faces. Our model will then predict the gender and age of the faces captured by the dlib library.

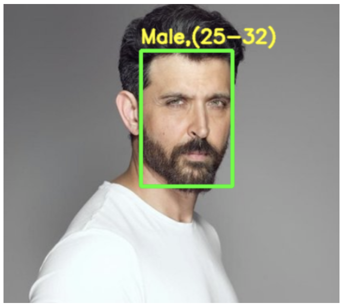


Figure 4.3.3 Prediction of Gender and Age

This is mainly possible due to EfficientNet's architecture which uses real-time inference, making it ideal for applications where immediate results are required. This is especially applicable for mobile devices which have limited computational resources, where EfficientNet's low memory footprint and computational costs makes it ideal for deployment on resource constrained platforms

5. Results & Discussion

5.1 Results

Comparing the metrics across our three models:

Models	Test Accuracy	Test Loss	Training Time
Levi Hassner	68.04%	0.5980	3958 sec
ResNet50	76.29%	0.4866	5441 sec
EfficientNetB0	**97.55%**	**0.0674**	**1830 sec**

Figure 5.1.1 Table of Comparison between all 3 models (Gender)

Models	Test Accuracy	Test Loss	Training Time
Levi Hassner	42.62%	1.5805	6914 sec
ResNet50	52.36%	1.5068	8464 sec
EfficientNetB0	**67.14%**	**0.8037**	**1830 secs**

Figure 5.1.2 Table of Comparison between all 3 models (Age)

5.2 Discussion

We draw a few key conclusions from our experimentation:

EfficientNet's Outstanding Performance

Comparatively, EfficientNet has outperformed the other trained models in both Gender and Age prediction after pre-training, hyperparameter tuning and adding additional layers. This can be attributed to two main factors:

- EfficientNetB0 was pre-trained on IMDB-Wiki Dataset with a subset of 60k images compared to 19k for Adience. This sheer volume of the available datasets granted to EfficientNetB0 allowed the model to learn more generalised features and patterns, prevent overfitting, better feature extraction and contributed to various other reasons critical for robust model performance
- EfficientNetB0 outperforms other models due to its unique compound scaling approach, systematically optimising model depth, width, and resolution for efficiency. The architecture's inverted residual blocks, mobile-friendly design, and parameter efficiency contribute to its

superiority. EfficientNet's adaptability to various tasks and strong performance across datasets make it a versatile choice, particularly in resource-constrained environments.

Levi-Hassner CNN vs ResNet-50: Efficiency & Performance

According to our implementation, Levi-Hassner performed better than ResNet-50 in computational time, whereas ResNet-50 performed better in its classification results in both age and gender. This is to be expected as the ResNet-50 model is a comparatively deeper architecture and would demand greater computational resources than the Levi-Hassner CNN. Hence, the trade-off between efficiency and performance for ResNet-50 is within expectations.

6. Further Improvements

As this research was developed as an assignment for a module, this project operates within a tight timeframe. Given the considerable computational resources and time investment required to process large datasets, we limited the training of the networks to 5 epochs, primarily for the purpose of establishing a basis for comparison between the models. For future research, practitioners may endeavour to increase the number of epochs for training to improve the performance of these models.

As demonstrated within this research, none of our proposed methods achieve an accuracy exceeding 70% for age prediction within 5 epochs. Hence, for the specific problem of age classification, it is worth considering the possibility that a linear regression approach might be more fitting than multiclass classification. The IMDb-Wiki dataset could be used with age labels from 0 to 102 instead of enumerated age groups.

While the Adience dataset initially includes a total of 26,580 images, our investigation revealed that due to labelling inaccuracies, only approximately 19,000 images were suitable for training. Given sufficient time, the exploration of a more extensive and comprehensive dataset beyond Adience could also be considered for future related research.

In this project, we have chosen EfficientNetB0 as our model for pretraining and transfer learning considering our time and resource limitations. The B0 variant is the smallest and least complex variant, and the model depth and number of parameters increase while going up the model. Larger variants perform better on computer vision tasks, and future projects might consider working with these aforementioned variants for future projects instead of the B0 variant.

References:

1. G. Levi and T. Hassner, “Age and gender classification using convolutional neural networks.” in IEEE Conf. on Computer Vision and Pattern Recognition (CVPR) workshops, 2015

2. Z. Liu and P. Luo and X. Wang, and X. Tang, “Deep learning face attributes in the wild,” in International Conference on Computer Vision (ICCV), 2015

3. He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (pp. 770-778).

4. A. Q. Bhatti and M. Umer, "Explicit Content Detection System: An Approach towards a Safe and Ethical Environment," in Hindawi, Applied Computational Intelligence and Soft Computing, vol. 2018, no. 2, Jul. 2018. DOI: 10.1155/2018/1463546.

5. M. Tan and Q. V. Le, “EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks” Proceedings of the International conference on machine learning, California, USA, June, 2019, pp. 6105–6114.