## Introduction

The goal of the project was to evaluate existing custom CNN models on Age classification and compare these models to a pre-trained model. The data used to train the models was based on the UTKFace dataset This data set consists of over 20,000 images of people ranging from age 0 - 116. We set out to compare the performances of the two models and made minor modifications to the Convolutional Neural network in order to improve it. The factor that mostly affected the performance of the CNN was changing the number of classes in the model.
As for the pretrained network we used the vgg16 model which is a very densely layered network and has pretrained weights, imagenet which is very useful in training face recognition algorithms.

## Individual Contribution

As part of this 2-person team, I took responsibility for choosing the dataset and scope of the project. I found the UTKFace data, and shared it with my partner. The initial idea behind the project was to use more than just the cropped faces available but there were also images of people “in the wild” i.e. pictures of people with more natural effects in the background. Some of these effects included landmarks that could have been used for a multi-label classification network. However, due to having trouble processing the images and encountering ‘fatal errors’ we were unable to preprocess the images for training. WI looked up previous/related work done on the “in the wild’” data but had no luck encountering anything that helped us. So we proceeded to use the cropped images for our model. Upon finding the CNN and inspecting it I decided that it would significantly improve the results by changing the age range of the classes we created, thereby increasing the number of classes to 6 from 5 in the reference model.

For the pretrained model, my partner helped build it and I was able to get it running on my system using the gpu which improved the speed but not the accuracy. On inspecting it, I noticed that the model was being attached to a sequential model which was not really a pretrained model so I adjusted the model that was created by calling the model output to the function x and using the global average pooling to downsample the images.

*x = vgg16_model.output*

*x = GlobalAveragePooling2D()(x)*

*predictions = Dense(6, activation='softmax')(x)*

*model = Model(inputs=vgg16_model.input, outputs=predictions)*

This allowed the pretrained model to run and improve the accuracy we got to over 80%.

## Model Results

| Models | Loss Function(on train) | Accuracy |
|---|---|---|
| Reference CNN | 0.8213 | 64.13% |
| Modified CNN | 0.6683 | 74.88% |
| Pretrained Network | 0.6885 | 83.62% |

Above are the results of the train script and due to a lack of a random seed the results fluctuate when you run the script multiple times.

## Summary and Conclusions

The evaluation confirmed already popular assumptions like more classes helps to improve accuracy and pre-trained models work better than models built from scratch.

## Code

Custom CNN: (100 -20)/(100+10) * 100 = 72.7%

Pretrained: (100-35) / (100+35) * 100 = 48.15%

## References


https://arxiv.org/abs/1702.08423

https://arxiv.org/pdf/1702.08423.pdf

https://machinelearningmastery.com/dropout-regularization-deep-learning-models-keras/

https://towardsdatascience.com/building-a-convolutional-neural-network-cnn-in-keras-329fbbadc5f5

http://cs231n.github.io/convolutional-networks/

https://www.tensorflow.org/tutorials/images/cnn

https://www.kaggle.com/sangarshanan/age-group-classification-with-cnn