CE/CZ4042 Neural Network and Deep Learning
Gender and Age Classification

We hereby declare that the attached group assignment has been researched, undertaken, completed, and submitted as a collective effort by the group members listed below. We have honoured the principles of academic integrity and have upheld Student Code of Academic Conduct in the completion of this work. We understand that if plagiarism is found in the assignment, then lower marks or no marks will be awarded for the assessed work.

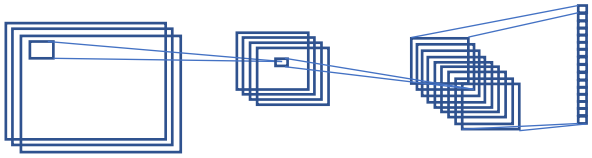
Name
Abhigyan Singh (U1922629G)
Asuri Simhakutty Kamakshi (U1923890E)
Das Atrik (U1823950C)

1. INTRODUCTION:

Under this section we will discuss the general concepts that have been used throughout the project:

1.1. Convolution Layers: In Deep learning and neural networks, we use many types of layers. One such layer is the convolutional layer. This layer, as the name suggests, performs the mathematical operation of convolution. Convolutions is a very common technique used in Computer vision. It is basically the technique to extract features from images using certain filters.

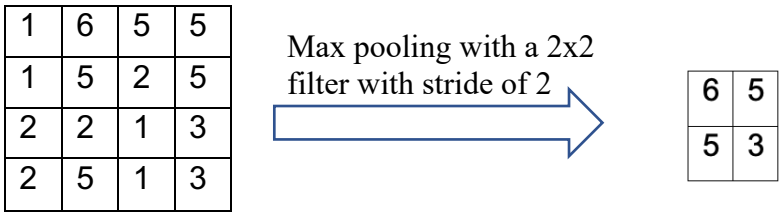
The design of these filters can vary for different purposes. In this project, we use the default Keras layer as we do not want to extract any feature from the image.
The purpose of convolution here is to convert a 3-d image (2 dimensions for pixels, 1 dimension for colours) into a 1-d array so that the image can be processed by the fully connected dense layers.



Convolution layers followed by Flatten layer

As we can see from the diagram above, convolution can be used to make the image suitable to be trained by the dense layers of the neural networks.
Some parameters that we must decide for the convolution layers include the number of filters, number of convolution layers, kernel/ filter size, padding size, stride (step taken by the filter while performing convolution).

1.2. Max Pooling Layer: This layer is usually used after a convolution layer. The purpose of this layer is to down-sample the input and hence extract some generalized features form the image.



1.3. Data Normalization: Normalization is a very important step in neural networks, as it allows us to level the range of inputs coming into the networks and so that there is no loss of information. It helps to provide a unform scale for all the different inputs coming into the network.
For this project we use Local Response normalization (LRN) rather than batch normalization. LRN is a very common technique used with convolution neural networks. It is a non-trainable layer that normalizes the pixel values in a local area. The advantage that LRN has over batch normalization is that LRN can normalize in multiple directions while batch normalization cannot.

1.4. Transfer Learning: This is a very useful technique which is used in neural networks. Transfer learning is the process of pre-training a model on a larger dataset, then freeze the layers of the model and then use the same model by adding an extra fully connected layer to train on out much smaller dataset. This method can be used when the size of our dataset is small and hence can help us to improve accuracies.

1.5. Adaptive Learning Rates: While training neural networks, it is always useful to decrease the learning rate as the training progresses. This is because we know that our loss function has a global minimum and as training progresses, we approach closer and closer to it. So, a lower learning rate at that time can prevent our training from oscillating and reach convergence faster.

But if we use the smaller learning from the very beginning, then the rate of convergence will become very slow. Hence we make use of adaptive learning rates, so that we can begin with a relatively larger learning rate and then slowly decrease it. There are 2 major techniques for adaptive learning rates:

1) Time based schedule
2) Step decay

For this project we will use step decay.

2. Project Motivation:

The Motivation of this project was to apply what we have learnt in the course CE/CZ4042 and use that knowledge to perform some real life tasks like gender and age classification. These application although seem very small but they have a huge impact in a fast growing industry of face and identity recognition. We hope to improve on the current project in the future to combine gender and age and even use it for some face recognition tasks.

3. Dataset Analysis:

In this Project we are using 3 datasets:

Adience Dataset: This is the dataset that has been used in [1]. This has a total of 26,580 images of around 2500 people. All these images are accompanied with specific labels for gender and age (7 different classes).

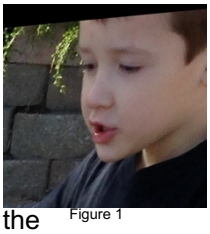


Figure 1

The images in this dataset have been cropped and pre-processed for gender classifications like cropping of the image region that includes the faces, aligning the images so that the images are straight etc.
To understand let's take two random images from the dataset. We can see here that the quality of the images is not the best, which will hinder the accuracy of the model. Also we can see that the alignment of figure 1 has already been done for us. This helps the model as we can focus more on the faces and hence train better.

Figure 2

On observing some more images we find that some of the images in the dataset have been misclassified i.e. some images of male gender have been given the label female and this will prove to be a hindrance while training. This dataset will be our main dataset to train on and get the test accuracies.

CelebA Dataset: This dataset is provided by TensorFlow and is an inbuilt dataset. It contains around 200,000 pictures of celebrity faces with the appropriate labels.

We can see from the sample images that the quality of these images are much better than the Adience Dataset. Since it has almost 10x the number of images that Adience had, it is much better for training. In this project, we use the CelebA dataset to pretrain our model and then use transfer learning to train the Adience dataset.

Wiki-IMDB Dataset: This is a dataset that we need in order to pre-train our model for age classification. We could not use the CelebA dataset as it doesn't have the respective age labels. This dataset is even larger than the CelebA dataset and has around 200,000 images with appropriate gender and age labels. But due to the limitation in RAM posed by the SCSE GPU we had to cut our dataset to only include around 90,000 images.
The quality of images of this dataset is again much better than the Adience dataset.

4. Pre-processing:

We perform pre-processing before any computer vision tasks as images can become very noisy and hence can hinder the training of the model. So, we use some techniques to make sure are image is compatible for training. The pre-processing can be as simple as just normalization of input or as complex as using canny filters for edge detection models. So, the pre-processing varies from the task that we want to perform.

Steps taken for the pre-processing of the images:

1) We read the text files that contain the paths of the images and the corresponding labels and create a new dataset with the paths and the labels that we need [gender and age]. We also shuffle all the values in the dataset to prevent any sequence bias from happening.
2) Then we assign numerical values to the labels which were earlier strings to make it easier for us to use.
3) Then we read the images and convert them into numpy arrays. After that we resize the images to 256x256. This was done as on observing the datasets we realized that many images had different sizes.
4) At this time, we also divide each and every pixel value by 256 to normalize the input and make sure that all the values of input images are in the range (0,1).
5) Then we just use list slicing to divide the data into test, train and validation split. We do not need to shuffle now as we already shuffled the dataset.

```
def df_to_list(df):
    data2 = []
    for i in range(len(df)):
        path = "/home/UG/abhigyan001/aligned/" + \
            str(df['user_id'][i])+"/landmark_aligned_face." + \
            str(df['face_id'][i])+"."+str(df['original_image'][i])
        if df['gender'][i] == 'm':
            gender = 0
        else:
            gender = 1

        if str(df['age'][i]) == str((0, 2)):
            age = 0
        elif str(df['age'][i]) == str((4, 6)):
            age = 1
        elif str(df['age'][i]) == str((8, 13)):
            age = 2
        elif str(df['age'][i]) == str((15, 20)):
            age = 3
        elif str(df['age'][i]) == str((25, 32)):
            age = 4
        elif str(df['age'][i]) == str((38, 43)):
            age = 5
        elif str(df['age'][i]) == str((48, 53)):
            age = 6
        elif str(df['age'][i]) == str((60, 100)):
            age = 7
        data2.append([path, gender, age])
    return data2
```

Figure 3: Function for converting the string labels to numerical values and make the new

```
for i in range(len(df_new)):
    image = Image.open(df_new['Path'][i])
    image.thumbnail((256, 256))
    data = asarray(image)
    for j in data:
        j = j/255
    df_image_x.append(data)
    df_gender_y.append(df_new['Gender'][i])
    df_age_y.append(df_new['Age'][i])
```

Figure 4: Code for reading image and performing pre-processing functions

```
def step_decay(epoch):
     init_lrate = 1e-3 #TOCHANGE
     drop = 0.1
     epochs_drop = 10000
     lrate = init_lrate * math.pow(drop, math.floor((1+epoch)/epochs_drop))
     return lrate
```

Figure 5: Function for adaptive learning rate using step decay

5. **Prevention of Overfitting:** In neural networks, overfitting is a common scenario as there are many parameters like weights, biases etc. Overfitting can be dangerous and decrease in accuracies. So, to avoid it need to use some techniques. The techniques we have used in the project to prevent overfitting:

    1) **Early Stopping:** Stop the training when the validation error begins to increase.
    2) **Dropout layers:** Train a fraction of weights in each epoch and in this we prevent overfitting by randomly dropping some neurons during training. The percentage of neurons to be dropped is given by the drop-out probability.

6. Models:

For this project, we are using the model mentioned in [1]. This model has 3 convolution layers followed by 2 Dense "ReLu" layers and finally a SoftMax layers performing binary classification for gender classification and multiclass classification for age. All the convolution layers are followed by a max pooling layer and a lambda [custom layer] which perform the function of local area normalization.

We can also see from the model diagram that each dense hidden layer is followed by a dropout layer with a dropout rate of 0.5 which is done to prevent overfitting.

We are using truncated normal weights initialization, with a mean of 0 and std dev of 0.01 as mentioned in the paper.

Also, we use step decay to have a dynamic learning rate that reduces after a set number of epochs. This helps is because as the number of epochs increases our step size keeps becoming smaller and smaller and hence helps us to reach the global minima much faster but at the same time during the early epochs where we are very far away from the global minima we have bigger steps using higher learning rates.

The optimizer we are using is SGD optimizer as mentioned by the paper with momentum which helps to improve convergence rate.

We will be performing various hyper parameter tuning experiments on the model mentioned in the research paper for both gender and age classification.
The experiment we will perform on both gender and age are:

1) Batch size
2) Number of neurons in the hidden ReLu layer
3) Number of hidden ReLu layers
4) Learning rate
5) Effect of data augmentation

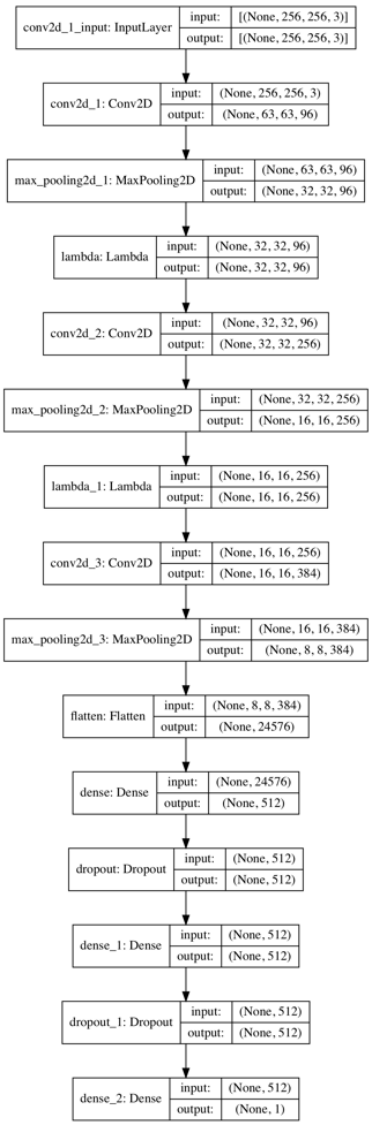


Figure 6: Model diagram

Apart from that we will be using this model and pre-train it using the CelebA for gender classification and Wiki-IMDB for age classification and make conclusions on the effect of using pretraining.

7. Experiments:

7.1. Optimal batch size for gender and age:

We try out different batch sizes [4,8,16,32,64] and see how it affects the accuracies. Batch size is an important parameter as it determines the accuracy of the determination of the error of the gradient.

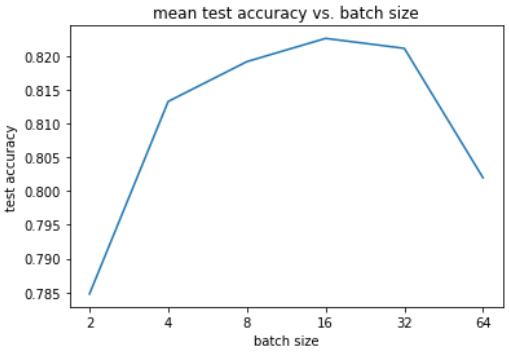


Figure 7: Test accuracies for various batches for gender classification

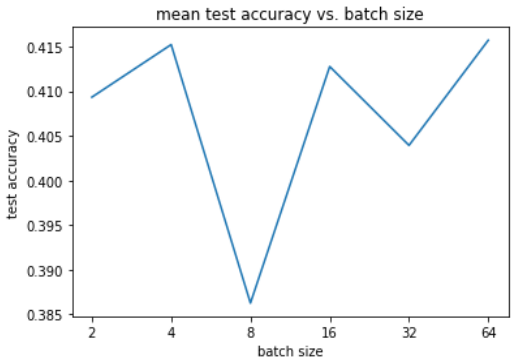


Figure 8: Test accuracies for various batches for age classification

Observations:

The graph here shows that the batch size of 16 is the best for gender and the batch size of 4 is best for age.

Also, it was observed that using a GPU if we train the model with a batch size of 1, we get NaN value for test accuracy if our leaning rate is less than 0.01. Our assumption as to why that happened was because we have a complex model and batch size of 1 means that each image is individually examined. And hence for individual images the loss->infinity. So, infinity in keras is shown by NaN.

7.2. Optimal number of neurons in the hidden ReLU layer:

The hidden ReLU layers at the end of the model are there to capture the non-linearity in the data. So, it is important to optimize the number of neurons in this layer. If the number of neurons in the layer is extremely high, it can increase the time to converge and hence increase the time to train.

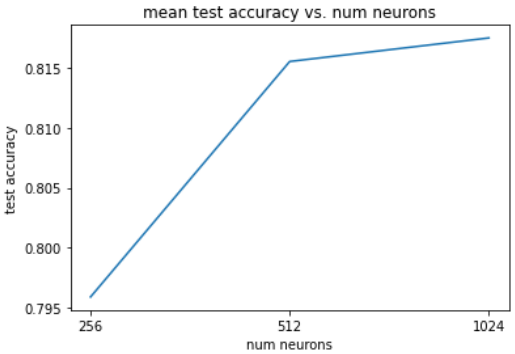


Figure 9: Test accuracies for various neuron numbers for gender classification

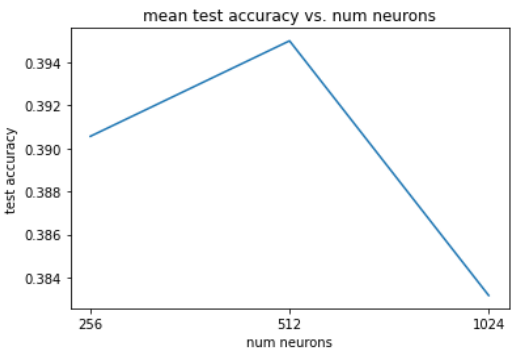


Figure 10: Test accuracies for various neuron numbers for age classification

Observations:

The graphs above show that the number of neurons 512 is the best for gender and the number of neurons 512 is best for age.

7.3. Optimal number of hidden ReLU layers:

Like optimizing the number of neurons, it is important to optimize the number of hidden ReLU layers so that they can accurately capture the non-linearity and at the same time does not add a lot of time to the training.

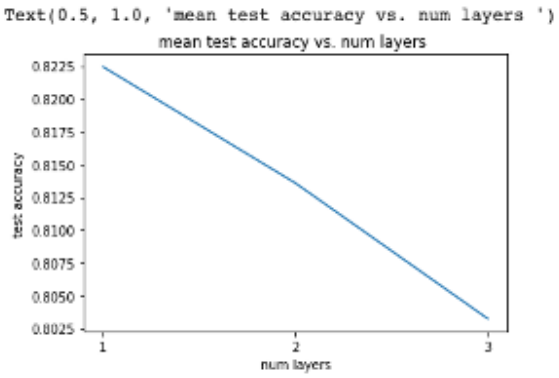


Figure 11: Test accuracies for various no of layers for gender classification

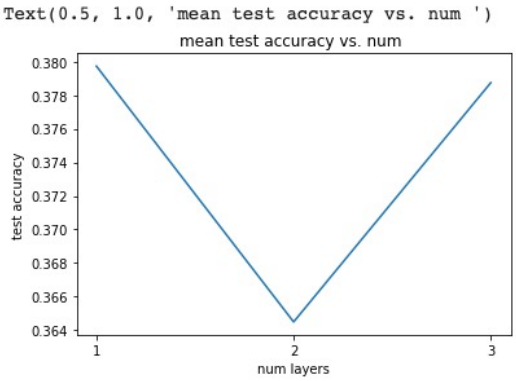


Figure 12: Test accuracies for various no of layers for age classification

Observations:

The graphs above show that the number of ReLU layers 1 is the best for gender and the number of ReLU layers 1 is best for age.

7.4. Optimal learning rate:

Although the research paper suggested using adaptive learning rate, we try to experiment to find out the ideal constant learning rate for the given dataset. The choice of learning rate is very crucial as a large learning rate can cause oscillations and prevent us from finding the minimum error while a small learning rate can lead to a high convergence time.

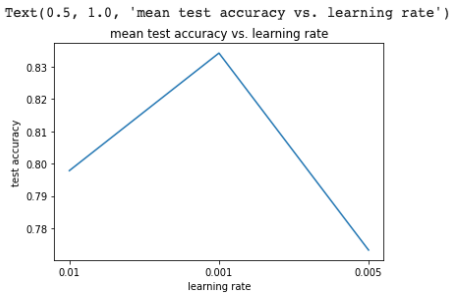


Figure 13: Test accuracies for various learning rates for gender classification

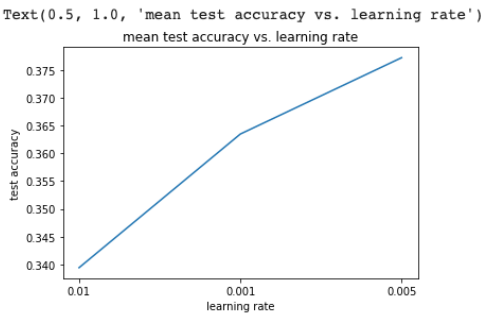


Figure 14: Test accuracies for various learning rates for age classification

Observations:

The graphs above show that the learning rate 0.001 is the best for gender and the learning rate 0.005 is best for age.

7.5. Effect of Data Augmentation:

All the experiments till now have not used any form of data augmentation. Data Augmentation is the technique used when the size of our dataset is small. Here since the Adience is a small dataset so we see the effect of a simple data augmentation technique of horizontal flipping on the accuracy.

With horizontal flip	Age	Gender
Test accuracy	0.42113021	0.85747798

Observations:

The table above show that even a simple data Augmentation technique like horizontal flipping can cause the accuracy of the model to increase. Hence, we can say that Data augmentation is a very effective technique to use when dealing with small datasets. We can see that it does not increase by a lot as we are dealing with face images and horizontal flip in faces might destroy the context of the image.

Note:

1) All the graphs above are for test accuracy only. For other graphs please refer to the analysis files attached with the report. Here you can find the validation graphs as well.
2) The accuracies of the experiments might seem to be lowering but that is only because the test set is not fixed and is randomly shuffled for each experiment. We can also notice that the change is not very steep, on an average all the accuracies are close to 0.4.

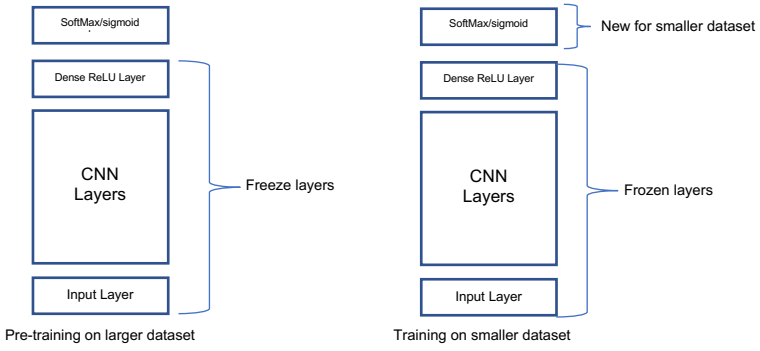
8. Transfer Learning:

8.1. Theory:

Another technique we tend to explore in this project is transfer learning. This technique again is used when the dataset is small like our Adience dataset. Here we pre-train our model on a larger dataset then we freeze all the layers of the model. Then, we add a fully connected layer on top and train it on our smaller dataset. In this way we are using the weight which are somewhat pre trained by the larger dataset and then fitting them to our dataset.

Steps for pre-training:

1) Train the model on a large dataset (CelebA, wiki-IMDB used here).
2) Save the model with the best val_accuracy during training.
3) Now load this model again, remove the last SoftMax layer and freeze all the other layers. By freezing the layers, we mean that the weights of these layers won't get trained.
4) Now add a new SoftMax layer and train this model on the smaller dataset (Adience here).



8.2. Experiments:

The following table has the test accuracies for transfer learning experiments:

Pre- Training	CelebA [gender]	Wiki-IMDB [Age]
Test accuracy	0.8634284	0.32383293

Observations:
From the table above we see that, the gender improved while the age didn't improve in fact the accuracy of age decreased. This can be explained by the fact that the IMDB and the Adience dataset are very different form one another. E.g. Adience had some pre-processing on it already done while IMDB did not.
Another reason is that due to the limitation posed by the GPU we could not run the full IMDB dataset and had to choose only 100,000 images which is even less than the CelebA dataset.

9. Review of Existing techniques:

1) In this project we followed the research paper proposed by Levi and Hassner. Most of the experiment results that we get align with their assumption but some hyper parameter tuning like number of layers is different.
2) We felt that the Adience dataset is not a very good dataset to use for training as the quality of some images is very bad and hence gets low accuracies.
3) Apart from that, some more data augmentation techniques can also be used as they can help to increase the size of the dataset and also make it more robust.


Figure 15: Example low quality image

10. Conclusion:

From the experiments conducted on age and gender classification, we get the following best models:

Parameters/model	Age	Gender
Batch Size	4	16
Number of Neurons	512	512
Learning rate	0.005	0.001
Number of hidden layers	1	1

From the data augmentation, we see that not all data augmentation techniques work for all the datasets. We should have good domain knowledge in order to find the best augmentation techniques for our dataset.

From the pre-training experiments, we learn the importance of similarity between the dataset being used for pre-training and the actual smaller dataset. We saw how pre-training can improve as well as reduce the accuracy for a given model. So it is very important to choose the datasets carefully.

11. Proposed Improvements:

1) From basic mathematics we know that convolution in the Fourier domain is multiplication, so using that instead of the convolution layer might make the model faster. With the availability of Fast Fourier transform [FFT], the time taken by FFT and inverse FFT combines should be less than the time taken by the convolution layers. So, instead of the convolution we would be doing this:

$$image_{convoluted} = invfft(fft(image) * fft(filter))$$

2) Due to the time constraint of this project, we had to run different experiments on different machines (SCSE GPU, google collab) which had different configurations and hence we could not make an inference or conduct various experiments comparing the times to train a batch or the time taken for an epoch. This can be done in the future and can help in the training of the model.

3) Since age can be represented as a continuous number (by using rounding up/down) we can also consider using linear regression for age prediction instead of using multiclass classification for age. This might help improve the accuracy as even in the dataset, wiki-IMDB all the ages were not grouped into classes/buckets and were numbers from 1-100. So instead of running a 100 class prediction we can run a linear regression model and predict the age.

4) Another improvement can be made by combining the age and gender models and have both of them complement each other. This would help improve the accuracy of both as we can take advantage of age-specific gender features and vice-versa.

5) The integration of computer vision into such a project can prove to be very useful. The reduction of noise using various gaussian filter or even thresholding via Otsu's algorithm can help us capture useful features in different parts of the face.

6) Use of much bigger and better quality datasets is needed specially for age classification.

12. Appendix:

10.1. 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] CE4042 NTU lecture notes.