Department of Computer Science

Submitted in part fulfilment for the degree of BEng

Age, gender and facial expression recognition using Convolutional Neural Networks

Vinusan Jeyananthan

30 June 2021

Supervisor: Dr Pengcheng Liu

ACKNOWLEDGEMENTS

I would like to express my thanks to Dr Pengcheng Liu for his guidance throughout the project. I would also like to thank those who have helped me proofread the paper for spelling and grammar mistakes.

STATEMENT OF ETHICS

To the best of my knowledge, there are no ethical, social, legal, commercial or professional issues related to the study in this project.

TABLE OF CONTENTS

Executive summary

The aim of this project is to produce three convolutional neural network (CNN) models capable of recognising age, gender and facial expressions when given an image of a face. Convolutional neural networks are a class of deep neural networks, often applied to images. They consist of convolutional layers that attempt to identify and extract features from an image. The models I am aiming to produce can have a variety of uses, from something as simple as an online age verification check, to enabling a robot companion to identify their owner's emotions and offer the appropriate support. However, in order to use them for further applications, I am aiming to achieve accuracies of 90%+. The secondary aim of my paper is to investigate the effects of tuning certain hyperparameters, altering the network architecture and using different optimizers, on the performance of CNN models built for age, gender and facial expression recognition.

Using the TensorFlow and Keras libraries offered in Python, I first imported both of my datasets and applied the correct formatting to extract the ground truth labels, i.e., what age / gender / facial expression an image was portraying. Next, I implemented the required functions for data augmentation – a popular process in machine learning, often used to add variation to the training data. As a result of the added variation, models often improve their ability to generalise and predict accurately on unseen data. In order to work towards models capable of high accuracies, I had to create a basic model to work up from. Having read through the relevant documentation and having watched the correct tutorials I proceeded to set up my simple CNN models for each recognition task. Once the base models were set up, I begun to tune various hyperparameters to see if I could achieve higher and higher accuracies. I have also tampered with the structure of each model's architecture and the optimizer functions (which define how the weights of a network should be adjusted to improve accuracy).

Having evaluated various configurations of the three CNN models against each other, I concluded on which setups produced the highest prediction accuracies when testing on unseen data. The finalised models gave me accuracies of 67.2%, 78% and 94.6% for age, gender and facial expression recognition, respectively. However, it was clear that better results were obtainable had I been able to access more powerful hardware, specifically a GPU with more memory.

Additionally, regarding my secondary aim, I discovered how ZCA whitening, one of many data augmentation techniques, slightly worsened my models' performances. On the other hand, using the Adam optimizer and implementing Dropout (a regularisation technique that aims to reduce overfitting) increased the performance of my models. I was able to successfully meet my secondary aim in understanding the effects of certain modifications I had made to the networks, in order to achieve higher prediction accuracies.

In future research, I would make sure to obtain access to more powerful hardware long before beginning my project. Additionally, although I have shown the effects of certain variations of the models, there are far more available for testing, including networks that make use of methods such as Inception modules or Batch Normalization.

1 Aim, Motivation and Approaches

The aim of this project is to create three high accuracy convolutional neural network (CNN) models: for age recognition, gender recognition and facial expression recognition. Although I will be aiming for as high as possible, the success criteria I have defined for this aim is as follows: 90%+ accuracy for the models' evaluations on their respective testing sets. A secondary aim is to investigate the effects of certain variations on the models' performance.

The motivation for this project stems from one key reason – the vast range of possible applications for a tool capable of recognising age, gender and facial expressions. These applications range from security-based uses such as with criminal identification through CCTV or biometrics, to human-computer interactions involving chatbots and robots. Since the rise of large, labelled datasets (e.g., ImageNet [14]), many advances have been made in the use of CNNs, to the extent where humans are regularly outperformed by models akin to the ones produced in this project.

In order to meet this success criteria, I have taken the following approach. Firstly, basic pre-processing steps were taken to import the datasets correctly and load them into a manipulable format. Next, a basic CNN architecture was implemented after looking into the necessary TensorFlow and Keras documentations. Finally, in order to push for models capable of achieving 90%+ accuracies, I have attempted a myriad of variations in pre-processing, data augmentation, hyperparameter tuning and regularisation techniques. Having completed this process, I arrived at 3 finalised CNN models for age, gender and facial expression recognition.

2 Literature Review

This section will begin by giving a brief overview of basic concepts, intending to give the reader enough information for full comprehension in later discussions. The section then proceeds to offer a critical summary and synthesis of the current literature on the use of CNNs for age, gender and facial expression recognition (FER).

2.1 Convolutional Neural Networks

Convolutional neural networks were first introduced by Yann LeCun in the 1980s [12], building on work done by Kunihiko Fukushima who, a few years earlier, had invented the neocognitron (a basic image recognition neural network) [13]. However, as CNNs required a lot of data and computational resources to work efficiently, they lacked scalability, only being applicable to low resolution images at the time. It was not until the creation of AlexNet in 2012 that deep learning was properly revisited [15]. The availability of large datasets, specifically the ImageNet dataset consisting of 3.2 million images [14], has since catapulted deep learning into a new age - allowing researchers to create complex CNNs capable of performing computer vision tasks that were once impossible.

A typical CNN architecture works in two stages: feature learning and classification. Feature learning is handled by 3 different layers: a convolutional layer, a non-linearity layer and a pooling layer.

2.1.1 Convolutional layer:

The convolutional layer applies filters over patches of the input image in order to produce feature maps. The filter performs element-wise multiplication and adds the outputs, sliding along the image by a set number of pixels (stride) until a complete feature map has been generated. A visual representation can be seen in Figure 1.

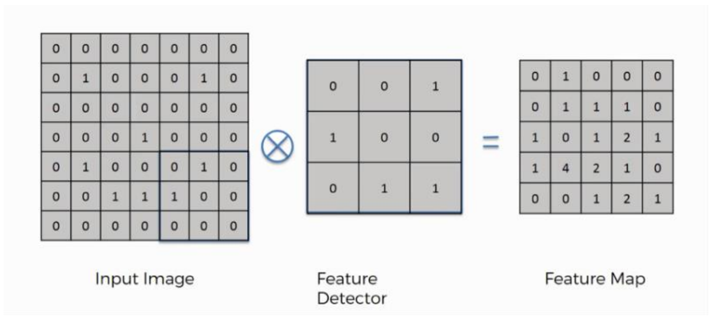


Fig. 1 - Convolution Operation

2.1.2 Non-linearity layer:

After each convolution operation, a non-linear activation function is applied to the output volume of the layer. This is done to account for the fact that image data is highly non-linear. The three most used activation functions are Sigmoid, Tanh and Rectified Linear Units (ReLUs). However, ReLUs are usually several times faster than their equivalents in training [15], thus tends to be the most popular choice. The ReLU function, as seen in Figure 2, outputs the input directly if it is positive, otherwise, it outputs zero.

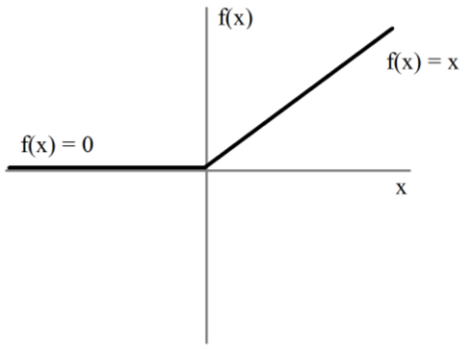


Fig. 2 ReLU function

2.1.3 Pooling layer:

The purpose of the pooling layer is to reduce the dimensionality of the inputs whilst preserving spatial invariance. One possible type of pooling is max pooling (depicted by Figure 3), where a filter slides across the image (akin to during convolution) and calculates the maximum value of the patch of pixels.

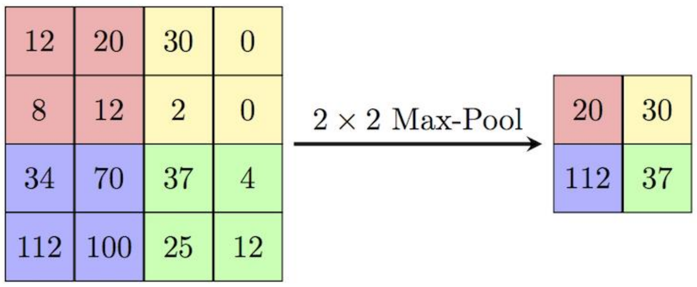


Fig. 3 - Max Pooling

Classification is performed by a fully connected layer and an output (loss) layer. The output from the convolutional layers represents high-level features in the image data. By adding a fully connected layer we provide a computationally cheap way of learning non-linear combinations of these features. Finally, the output layer uses a loss function on the input from the fully connected layer, in order to compute the error in prediction.

2.2 Data Pre-processing

“Garbage in, garbage out.” An old machine learning lesson that reminds us - without high quality input data, even the best CNN architectures will not be capable of performing well. Pre-processing is a step we take to ensure that our image data is clean and correctly formatted. For example, inputs into fully connected layers of a CNN must be of the same dimensions. Additionally, if the raw data contains large images, reducing their size may improve model training time without sacrificing model performance.

Two of the most well-known pre-processing techniques are Mean Normalisation and Standardisation. Mean Normalisation consists of computing the mean along each of the features of training samples and subtracting it from each image, whilst Standardisation consists of first performing Mean Normalisation – then dividing by the standard deviation along each of the features of training samples. Both techniques allow CNNs to achieve better performance during image classification than using raw image data (illustrated by Figure 4. However, Zero Component Analysis Whitening (ZCA whitening) has been shown to comfortably outperform Mean Normalisation (with an 8% increase in accuracy) and edges past Standardisation (with a 1% increase in accuracy) [16].

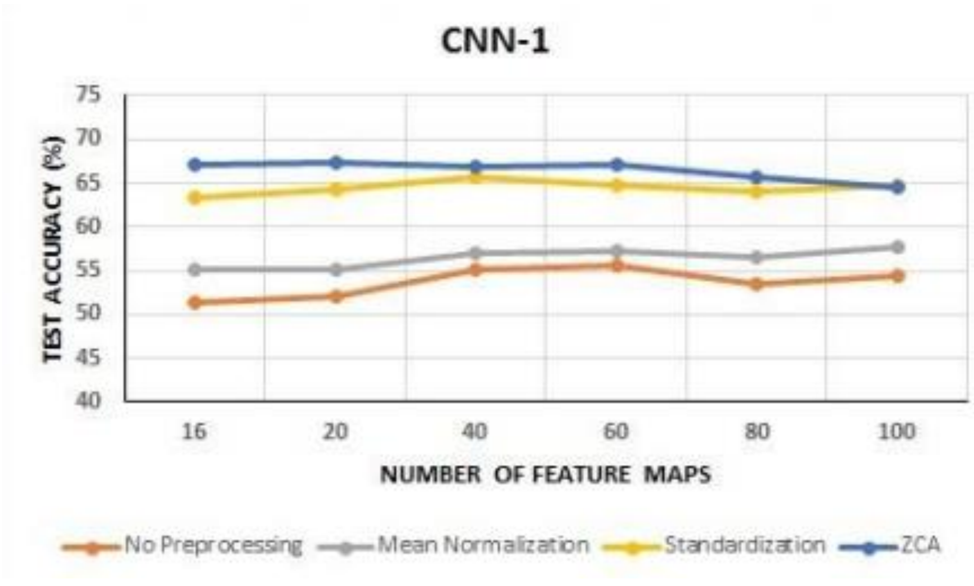


Fig. 4 - Test accuracy vs Number of Convolutional layer Feature Maps [16]

The ZCA whitening transformation was first tested by Krizhevsky to force a statistical model to focus on higher-order correlations than strong correlations between nearby pixels [21]. After whitening it is not possible to use the value of only one pixel to predict the value of another pixel – removing the distractions of two-way correlations and allowing the model to discover more abstract regularities between images.

2.3 Augmentation

A common problem in image-based deep learning tasks is overfitting when we do not have a sufficiently large and varied dataset. Thus, augmentation is often used to artificially expand the size of a training dataset by creating altered versions of already existing images. Augmentation allows us to provide our model with more images and greater variation across them – improving its ability to generalise and boosting the model's accuracy. Although there is a multitude of different augmentation techniques, they can generally be sorted into two categories: data warping and oversampling [19].

2.3.1 Data Warping

Data warping augmentations work by performing transformations on existing images whilst preserving their label. This includes geometric and colour transformations, neural style transfer and random erasing.

An example of a commonly used geometric transformation is horizontal axis flipping. Its popularity stems from being one of the easiest augmentations to implement. Cropping, rotation and translation are also used frequently.

Noise injection using a matrix of random values taken from a Gaussian distribution has also been shown to lead to better prediction accuracies and accelerated training performance in CNNs [22].

2.3.2 Oversampling

Oversampling augmentations create new images synthesised from existing ones and add them to the training set. Some examples are generative adversarial networks (GANs), feature space augmentations and mixing images.

GANs are an example of generative modelling: the process of creating artificial images from a dataset that have similar characteristics to the original images. However, the vanilla GAN architecture first proposed by Ian Goodfellow [23] fails to produce quality results for more complex datasets with higher resolution. This prompted the creation of CycleGAN [24], which has since been used for an Emotion Classification task by Zhu et al. [25]. In this work, CycleGAN succeeds in creating imbalanced classes with images from the FER2013 dataset [26], by translating images from one emotion class to another (see Figure 5).

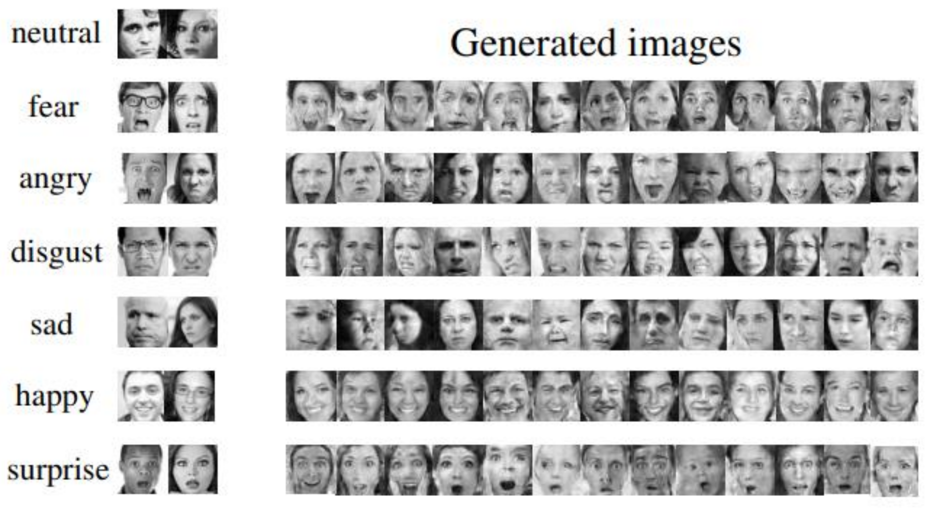


Fig. 5 - The left two columns are original data, whilst the rest are generated by CycleGAN [24]

2.4 Age Recognition

The task of age classification has been approached from various angles over the years. Past approaches were often based on finding ratios between measurements of facial features [28] [29]. Ratios were calculated by locating key facial features (nose, ears, eyes etc.) and measuring their sizes and distances between each other. These ratios were then used to sort images into different age categories using some set of handmade rules. However, as this approach requires accurate localisation of facial features, it proves to be ill-suited for in-the-wild, unconstrained images.

As GPU resources have become more powerful over the last decade, deep CNNs have become an increasingly employable tactic for age classification. Nevertheless, as of 2015 a large issue remained – finding a large enough dataset to prevent overfitting. Creating a labelled dataset for age estimation requires access to large amounts of personal information and as a result, there was a lack of a dataset matching the size of ImageNet [14]. The issue of overfitting is especially amplified in the case of deep CNNs due to their vast numbers of model parameters.

One approach that reduces the risk of overfitting is choosing a smaller network design [4]. Their network comprised of 3 convolutional layers and 2 fully connected layers with a small number of neurons. Additionally, using a smaller network was more appropriate for their task of distinguishing between eight classes on the Adience dataset [31] (each class being an age group) compared to other work that used thousands of classes to train a network [30]. An important distinction with Levi and Hassner's model [4] is that their model was not pre-trained with any data outside of the Adience benchmark - compared to models such as DEX [27], which used a VGG-16 architecture pretrained on ImageNet, and DAGER [1], where their base model is trained on over 4 million images of more than 40,000 individuals.

A key finding between multiple papers was the impact of face alignment on the performance of their models. Misalignments were reported to have a noticeably negative impact on the performance of models in [4] and [1]. In order to combat this effect, [4] implemented over-sampling by extracting 5 regions from the images, 4 from the corners of the face and the 5th from the centre. Their network was then presented with all 5 regions and their horizontal reflections, with the final prediction value taken as the average prediction value across each of the variations. A different solution was used in [27], where they ran a face detector [32] on the original image and on rotated versions. The version with the highest detection score was taken and rotated to a frontal position, before being used for prediction (see Figure 6).

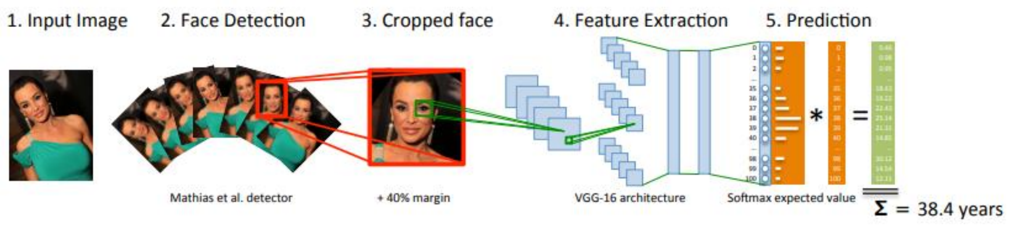


Fig 6. - Pipeline of DEX for apparent age estimation [27]

2.5 Gender Recognition

Many of the early approaches to gender recognition did not employ the use of CNNs. Instead, they used methods such as raw image pixels with nonlinear support vector machines (SVMs) [33] or local region-based feature extraction with SVMs [34] to achieve accuracies of 96.62% and 94.2% respectively. However, these studies only considered images taken under controlled conditions (e.g., FERET database). As these images were often frontal, with consistent lighting and clean backgrounds, these studies failed to address the difficulties of gender recognition in real-world scenarios. When images are taken of real-life faces there is often significant variation in facial poses, lighting, make-up / cosmetics and facial expressions. This makes gender recognition with real life faces far more challenging than for faces captured in constrained conditions. In order to test gender recognition on real-world faces, the Labelled Faces in the Wild (LFW) [35] dataset was built (see Figure 7) and has since been used as a benchmark for gender recognition across multiple studies.

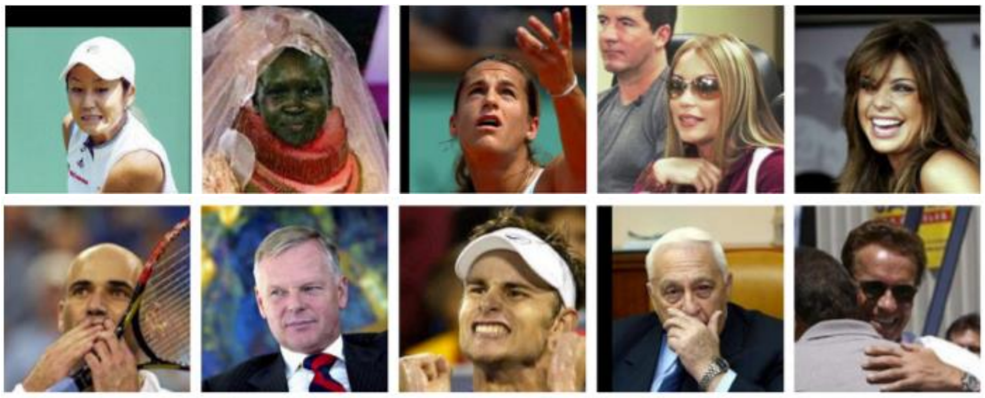


Fig. 7 - Examples of images from the LFW database taken in unconstrained conditions [6]

One of the more recent studies [7] that tackled the problem of gender recognition in unconstrained conditions opted to use a compact Deep

Convolutional Neural Network (DCNN). Additionally, the researchers wanted to find a method that would be able to run in real time on embedded devices. As a result, they succeeded in achieving state-of-the-art accuracy whilst highly reducing computational cost. Greco et al. selected a pre-existing architecture known to produce high accuracies in image classification tasks (MobileNets v2 [36]) and optimised for reduced latency by reducing input sizes, the number of layers and the number of feature maps. Their proposed architecture achieved up to 98.73% accuracy on the LFW dataset, even without the use of a dedicated GPU. Thus, this method can be extended to applications in CCTV cameras or commercial robotics, where often there is only a low power CPU to make use of, without sacrificing accuracy.

2.6 Facial Expression Recognition

Traditional methods of tackling facial expression recognition are based on engineered features such as Local Binary Patterns (LBP) [37] or Histogram of Oriented Gradients (HoG) [38]. However, these approaches lack in generalisability due to using classifiers with hyperparameters tuned to give high accuracies for a single dataset, or a small set of similar datasets. Consequently, when applied to novel datasets, these approaches perform poorly.

In order to work around this issue, Mollahosseini et al. [10] worked on a DCNN-based solution inspired by AlexNet's success in achieving a top=5 error rate of 15.3% on the ILSVRC-2012 competition [39]. The aim of their research was to propose an architecture comparable to or better than state-of-the-art methods for FER in accuracy and training time. Increasing the number of neurons or layers in a neural network architecture has often been used as a technique to improve accuracy as it can allow a network to learn more complex functions. However, the increased depth and complexity of the network can lead to overfitting and increased computational costs. To remedy this concern, Mollahosseini et al. proposed the application of an Inception layer shown to have remarkable success in face recognition [40], of which FER is a subdomain. The proposed architecture achieved a 66.4% Top-1 accuracy on the FER2013 dataset [41], beating a standard benchmark AlexNet's 61.1%. Furthermore, AlexNet performed more than 100M operations whilst Mollahosseini et al.'s network performed ~25M operations. The Inception layers had succeeded in increasing the network's depth and width whilst keeping computational costs low.

3 Design and Implementation

This chapter aims to provide a discussion on various factors taken into account during the design of this project including software and hardware choices, datasets, loss functions, optimizers and batch sizes.

3.1 Experimental Setup

Due to restrictions caused by COVID-19, I was unable to easily access powerful computers offered by the University of York computer science department. Certain decisions discussed later in this paper will reflect this issue. I have instead used my personal PC for my research, which is equipped with an i5-4690 CPU, an Nvidia GTX 970 GPU and 16GB of ram. Fortunately, the GTX 970 is CUDA (Compute Unified Device Architecture) – enabled, speeding up the process of training models.

In order to conduct my research, I have opted to use TensorFlow [44] and Keras [45]. TensorFlow is an open-source machine learning library for Python that allows for abstraction from the mathematical details of implementing algorithms. Keras runs on top of TensorFlow and reduces the cognitive load for developers by offering simple but consistent APIs. Together with the availability of extensive documentation and video tutorials, picking TensorFlow and Keras for the implementation of my neural networks was an easy choice. Additionally, Keras offers various pre-processing utilities [46] for manipulating datasets.

3.2 Data sets

For age and gender recognition, I have chosen to use the easily accessible UTKFace Dataset [43] created by Yang Song and Zhifei Zhang. UTKFace is a large-scale dataset consisting of 20,000+ aligned and cropped JPEG images of size 200 x 200. Being able to use the aligned and cropped images removes the extra complication of face detection before being able to tamper with different models. The images cover an age span of 0 to 116 years old, and images vary greatly in illumination, occlusion, facial expression and pose. The images have been annotated with the ground truth ages, genders and ethnicities. These features make UTKFace applicable for a variety of facial recognition-based tasks including research into age progression / regression by conditional adversarial autoencoders [47] and in my case of age and gender recognition.

For facial expression recognition, I have opted to use The Extended Cohn-Kanade Dataset (CK+) [42] created by Jeffrey Cohn. The CK dataset has been one of the most widely used datasets in detection facial expressions since 2000. However, one key limitation in the initial CK dataset was the lack of validation for emotion labels. The labels were documented as what expression was requested of the subject in the image, instead of what was actually performed. Furthermore, unless a quantifiable metric for determining the labels' validities is created, it is impossible to compare human standards against an expression recognition algorithm's performance. In order to fill these gaps, a selection process was introduced in which the images were relabelled according to the Facial Action Coding System (FACS) codes and Action Units (AU) [57]. The refined CK (CK+) dataset was released in 2010. The available version that I could access (CK+48 cropped) consists of 900+ black and white PNG images of size 48 x 48, spanning across 7 emotions: anger, contempt, disgust, fear, happy, sadness and surprise.

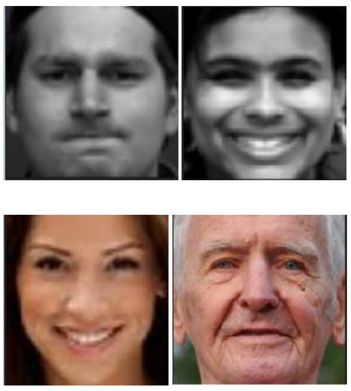


Fig.8 - Examples of the CK+48 Dataset (above) and the UTKFace dataset (below)

3.3 Loss functions

Whilst a model is being trained, the values of the weights of the network are adjusted through backpropogation, in order to minimise the error between the ground truth and the predicted output. Loss functions are responsible for computing this error, thus enabling the model to adjust weights accordingly.

Age, gender and facial expression recognition are tasks that fall under the category of multi-class classification, in which typically categorical cross-entropy loss is used [48]. The categorical cross-entropy loss function is calculated through the following sum:

$$\text{Loss} = -\sum_{i=1}^{\substack{\text{output}\\ \text{size}}} y_i \cdot \log \hat{y}_i$$

Fig.9 – Categorical cross entropy loss function

Output size is the number of scalar values in the model's output, whilst y_i is the target value and $\hat{y}_i$ is the i-th scalar value in the model's output. The purpose of the minus sign is so that the loss decreases as the probability distributions for y_i and $\hat{y}_i$ get closer to each other.

3.4 Optimizers

Optimizers are algorithms that define how the weights or learning rates of a neural network should be changed to minimise loss – thus producing more accurate results. The most basic optimization algorithm is gradient descent, which is an iterative algorithm dependent on the first order derivative of the loss function. It computes the direction in which the weights should be altered such that the function may reach a minima (see Figure 10).

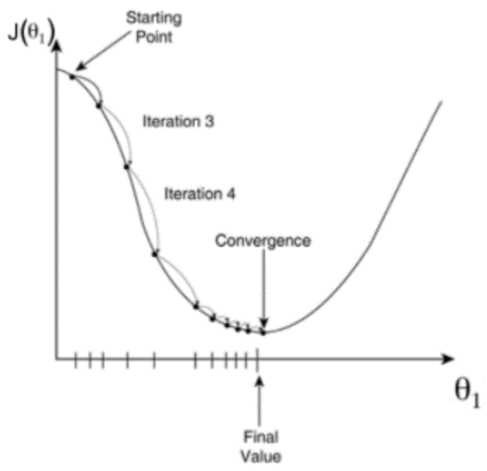


Fig. 10 – Visual representation of the stepping process in a gradient descent algorithm

Though gradient descent offers advantages in its easy computation, it is prone to becoming trapped at local minima, thus failing to find the true global minimum of the loss function. Additionally, as weights are adjusted after computing gradient across the whole dataset, a large dataset can cause increases in the computational time taken for convergence. Stochastic Gradient Descent (SGD) attempts to resolve this issue through updating the model’s parameters after each training example, in our case, after each image. Though this approach can result in a quicker convergence, it also leads to high variance within the model parameters themselves.

A key issue that arises when using any variant of gradient descent is the need to choose an optimum value for the model’s learning rate. Failing to pick a large enough learning rate can lead to gradient descent taking too long to converge. Attempts to solve this complication brought about a momentum term, which accelerates the convergence towards the correct direction whilst reducing any movements towards the wrong direction caused by the high variance in SGD. One algorithm that uses momentum is Adaptive Moment Estimation (Adam) [49], which converges much quicker than SGD and removes the need to find an optimum learning rate as it self-tunes the individual learning rates for each parameter and does so at each time step [50]. Although in most situations Adam outperforms other optimization algorithms, it often comes at the cost of higher computational costs. Moreover, in the case that the optimum values for the hyperparameters have been found, SGD can achieve better results than Adam.

3.5 Batch size

Batch size is another hyperparameter that needs to be carefully considered when creating neural network models. In our case it will be defining the number of images that the model will process before updating its parameters. Consequently, the batch size can have an effect on the rate of the network’s convergence. Using a smaller batch size requires less memory and often the model will take less time to train. On the other hand, using larger batch sizes can lead to higher image recognition accuracy at the cost of increased computational costs [51].

4 Implementation

The aim of this chapter is to outline and justify the low-level implementation details for each model and discuss the properties of each network that have been varied to give the highest accuracies.

4.1 Age Detection Model

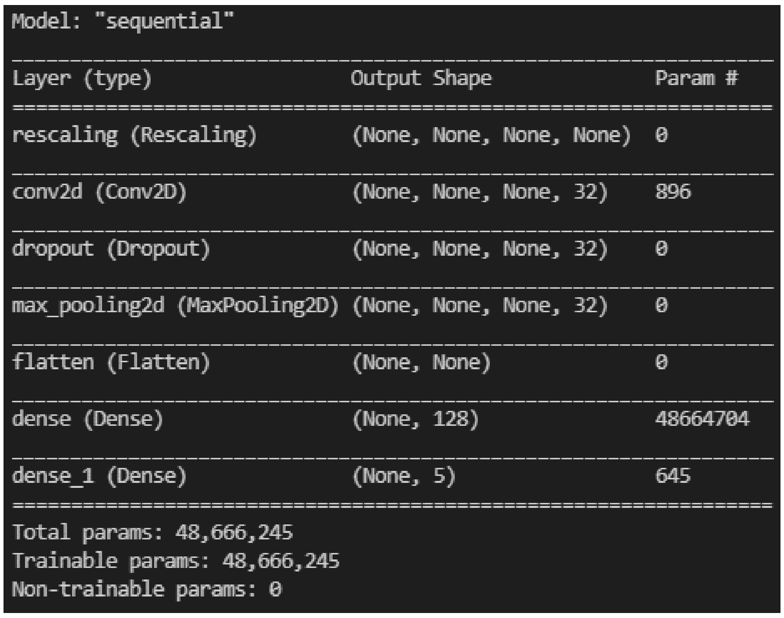
In order to implement the age detection model, I first needed to correctly import and pre-process the images and their labels. Firstly, I created a for loop structure that took in the path names for the images and split it up as needed to obtain the ages for each image. Next, to turn this problem into a multi-class classification task, I proceeded to use an if statement to sort the age labels into age ranges of 0-20, 21-40, 41-60, 61-80 and 81-116. Now the number of classes the model must predict from is 5. Once these age range labels were created, I appended the image and age columns into a pandas.DataFrame [52]. DataFrames are a similar structure to SQL tables or Excel spreadsheets but are more efficient and easier to use, being integrated into Python and offering vast functionality in manipulating columns, rows or the values within.

Next, I needed to split the data up into training, testing and validation data. To create these subsets, I used scikit-learn’s train_test_split function [53] to first create an 80/20 split of training and testing data – to set aside 20% of the dataset for final testing. Then I split up the training data from the first split with an 80/20 split to separate into training and validation data. The validation set will be used to build the model, being used for the model’s parameter selection and to reduce overfitting. It is used to tune the internal parameters of the model. On the contrary, the testing set is used to evaluate the model’s performance after it has been built. Additionally, the image data is rescaled by multiplying the pixel values by 1/255.

As discussed earlier, augmentation is an important process that increases the variation in the dataset, aiming to reduce overfitting and increase the model’s ability to generalise for unseen data. Using Keras’ ImageDataGenerator function [54], I was able to generate additional batches of image data with real-time data augmentation. After trying various configurations, I found that for the age detection model, best results were achieved when using a horizontal flip and a rotation range of 45°. Horizontal flips have been shown to potentially aid in remedying issues caused by uneven illumination across the face.

For all three of my models, I have decided to use a batch size of 32. Although I tried using other often used batch sizes of 64 and 128, they resulted in long training times making it impossible to repeatedly test the models with different parameters / network architectures.

In order to construct the model, I have opted to use Keras' Sequential architecture. Through trial and error of various network architecture configurations, I settled on using 4 convolutional layers with ReLU activation functions stacked together: each of them consisting of 32 filters and a kernel size of 3 x 3. Each convolutional layer has a default stride of 1 x 1 and each layer is followed by a 20% Dropout and a 2 x 2 max pooling layer. Increasing the number of convolutional layers further did not result in a better performance and only added to the complexity and computational costs of the network – thus I have only used 4. The output from the final convolutional layer is then flattened and fed into the fully connected layer with a ReLU activation function, then into an output layer with 5 outputs – one for each age group class. For the optimization algorithm I have chosen Adam optimizer for all 3 models as, in addition to previously discussed reasons, it is widely considered to be the most robust choice and requires little configuration – the default configuration parameters perform well (learning rate of 0.001 and initial exponential decay rates of 0.9 and 0.999 for the 1st and 2nd moment gradient estimates respectively). Additionally, it is highly computationally efficient and has little memory requirement making it appropriate for the limited GPU power from the hardware I am using. The loss function used was TensorFlow's SparseCategoricalCrossentropy instead of the standard CategoricalCrossentropy as the way I have loaded the dataset configures the true labels as integers representing class indices (e.g., 1 = ages 0-20, 2 = ages 21-40 etc.).



```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
rescaling (Rescaling)        (None, None, None, None)  0
_________________________________________________________________
conv2d (Conv2D)              (None, None, None, 32)    896
_________________________________________________________________
dropout (Dropout)            (None, None, None, 32)    0
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, None, None, 32)    0
_________________________________________________________________
flatten (Flatten)            (None, None)              0
_________________________________________________________________
dense (Dense)                (None, 128)               48664704
_________________________________________________________________
dense_1 (Dense)              (None, 5)                 645
=================================================================
Total params: 48,666,245
Trainable params: 48,666,245
Non-trainable params: 0
```

Fig. 11 – Age model summary

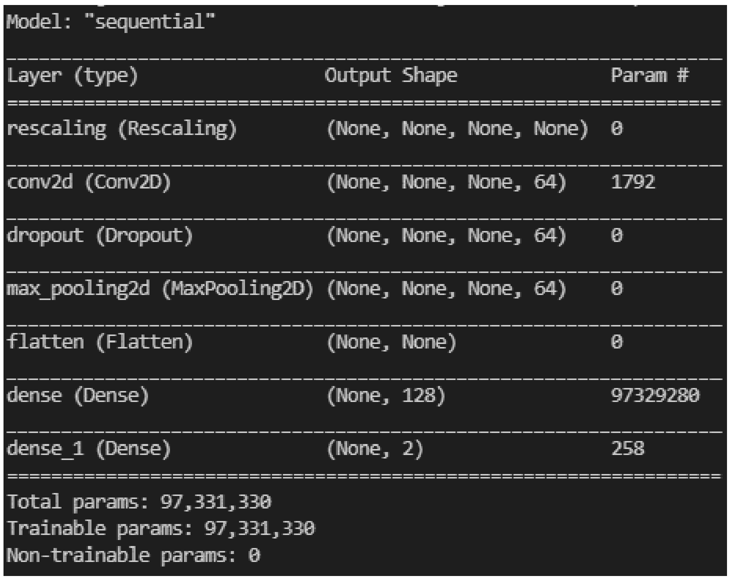
When training a model there can often be a point where the model stops generalising and begins to improve its ability to detect small fluctuations in the training set data, i.e., the model begins to overfit to the training set. One method that battles this effect is called Early Stopping, a regularisation technique that works by monitoring the model's performance during training and triggering a halt when some criteria has been met. I have implemented Early Stopping for all three models using Keras' Callbacks API [55] to monitor the validation loss during training. By setting the EarlyStopping class's patience parameter to 5, the models will now stop training if 5 epochs have passed in which there has been no improvement in the validation loss – thus reducing the chance / amount that the models overfit.

4.2 Gender Detection Model

As I am using the same dataset for gender detection as for age detection, the pre-processing steps are mostly the same as I have already discussed in the previous section. An if loop has been used to interpret the labels as '1' = female and '0' = male instead of age classes, but the same process has been used for splitting training, testing and validation data as in the age detection model.

Regarding augmentation, a horizontal flip and rotation range of 45° has been implemented, similarly to with the age detection model. However, inspired by earlier discussed literature, I have also attempted to find success with ZCA whitening.

After experimenting with different configurations of network architecture, I was presented with the highest accuracies when using 6 convolutional layers with ReLU activation functions. Each convolutional layer consisted of 64 filters and kernel size of 3 x 3, using the default stride of 1 x 1. Additionally, each layer is separated by 20% Dropout and 2 x 2 max pooling layers, in similar fashion to the age detection model. Finally, the output from the 6th convolutional layer is flattened and fed into a dense layer with ReLU activation function, then into an output layer with 2 outputs – one for male and one for female. In contrast to the age detection model, the gender ground truths are one-hot encoded, prompting the use of CategoricalCrossentropy over SparseCategoricalCrossentropy.


```
Model: "sequential"
_________________________________________________________________
Layer (type)                Output Shape              Param #
=================================================================
rescaling (Rescaling)       (None, None, None, None)  0

_________________________________________________________________
conv2d (Conv2D)             (None, None, None, 64)    1792

_________________________________________________________________
dropout (Dropout)           (None, None, None, 64)    0

_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, None, None, 64)   0

_________________________________________________________________
flatten (Flatten)           (None, None)              0

_________________________________________________________________
dense (Dense)               (None, 128)               97329280

_________________________________________________________________
dense_1 (Dense)             (None, 2)                 258

=================================================================
Total params: 97,331,330
Trainable params: 97,331,330
Non-trainable params: 0
```

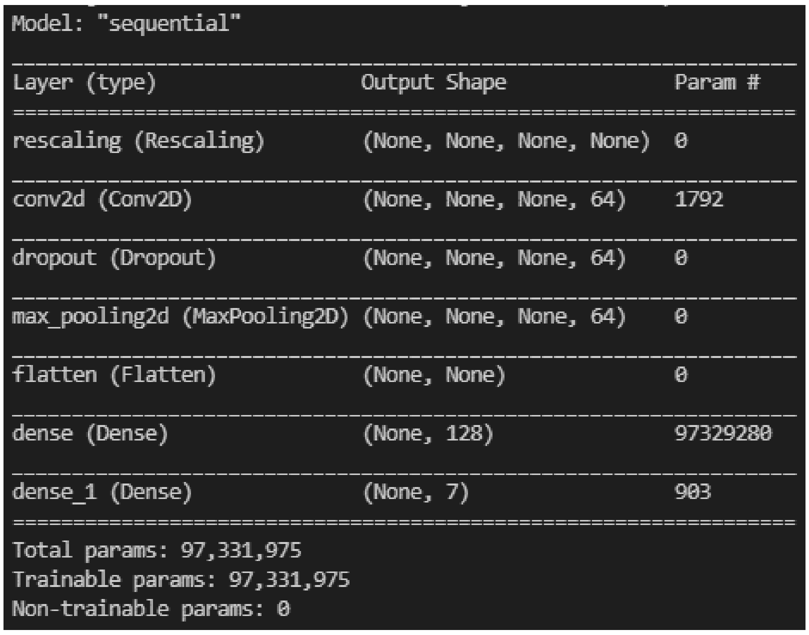
Fig. 12 – Gender model summary

4.3 Facial Expression Model

The method of pre-processing I have used is largely the same as for age and gender, except the labels have been sorted into 7 emotion classes: 'anger', 'contempt', 'disgust', 'fear', 'happy', 'sadness' and 'surprise'. The training, testing and validation sets have also been created in similar fashion, with two 80/20 splits.

Emotion recognition is a harder problem than age and gender recognition as there are now 7 classes for the network to distinguish between. Thus, I have opted for a more complex network architecture – stacking 7 convolutional layers and adding more parameters to the model. Each layer consists of 64 filters and uses a kernel size of 3 x 3, followed by a 20% Dropout and 2 x 2 max pooling layer. The output is then flattened and sent through to a fully-connected layer, and an output layer with 7 outputs – one for each emotion class.

Due to the CK+48 dataset being smaller in size than the UTKFace dataset used for age and gender detection, I have been able to test the FER model with a higher number of epochs (50-100), leading to higher overall accuracies than with age and gender detection.



```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
rescaling (Rescaling)        (None, None, None, None)  0
_________________________________________________________________
conv2d (Conv2D)              (None, None, None, 64)    1792
_________________________________________________________________
dropout (Dropout)            (None, None, None, 64)    0
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, None, None, 64)    0
_________________________________________________________________
flatten (Flatten)            (None, None)              0
_________________________________________________________________
dense (Dense)                (None, 128)               97329280
_________________________________________________________________
dense_1 (Dense)              (None, 7)                 903
=================================================================
Total params: 97,331,975
Trainable params: 97,331,975
Non-trainable params: 0
```

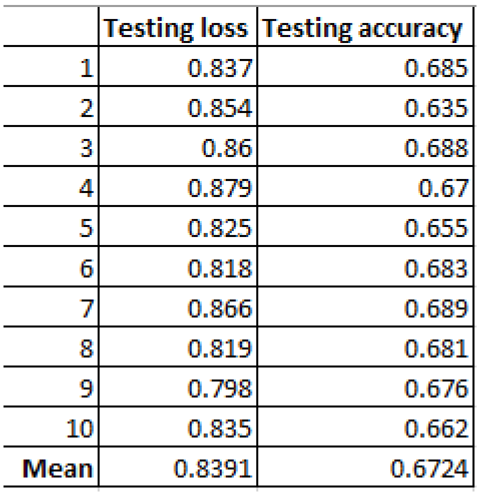
Fig. 13 – FER model summary

5 Results and Discussion

In this chapter, I will firstly show and discuss the results achieved using the finalised models as described in Section 4. Next, I will outline the variations of the models I tested before settling on the final versions, through changing the network architecture, optimizers and implementing regularisation techniques.

5.1 Results of age model

The evaluation metric used for all three models is the standard Accuracy class from Keras [58], which simply calculates how often a model's predictions match the ground truth. In order to calculate the mean value for this model's loss and accuracy when evaluating on the testing set, I recorded results for 10 runs (to 3 significant figures) and calculated the arithmetic mean. The loss is an output of the loss function – we want it to be get as close to 0 as possible. However, it must be noted that due to GPU memory limitations, the size of the dataset caused training times to be too high to run the model repeatedly at a high number of epochs. Thus, the following results have been achieved using just 20 epochs.



	Testing loss	Testing accuracy
1	0.837	0.685
2	0.854	0.635
3	0.86	0.688
4	0.879	0.67
5	0.825	0.655
6	0.818	0.683
7	0.866	0.689
8	0.819	0.681
9	0.798	0.676
10	0.835	0.662
Mean	0.8391	0.6724

Fig. 14 – Age model results

The final age model achieved some successful results, with a mean loss of 0.839 and a mean Top-1 accuracy of 67.2%. Considering the fairly shallow architecture and only training for 20 epochs, it is surprising that the model still manages to reach 67.2% accuracy. Additionally, in order to visualise the model's training history, I have used pyplot to plot two graphs, one showing the training and validation accuracy and one showing the training and validation loss.

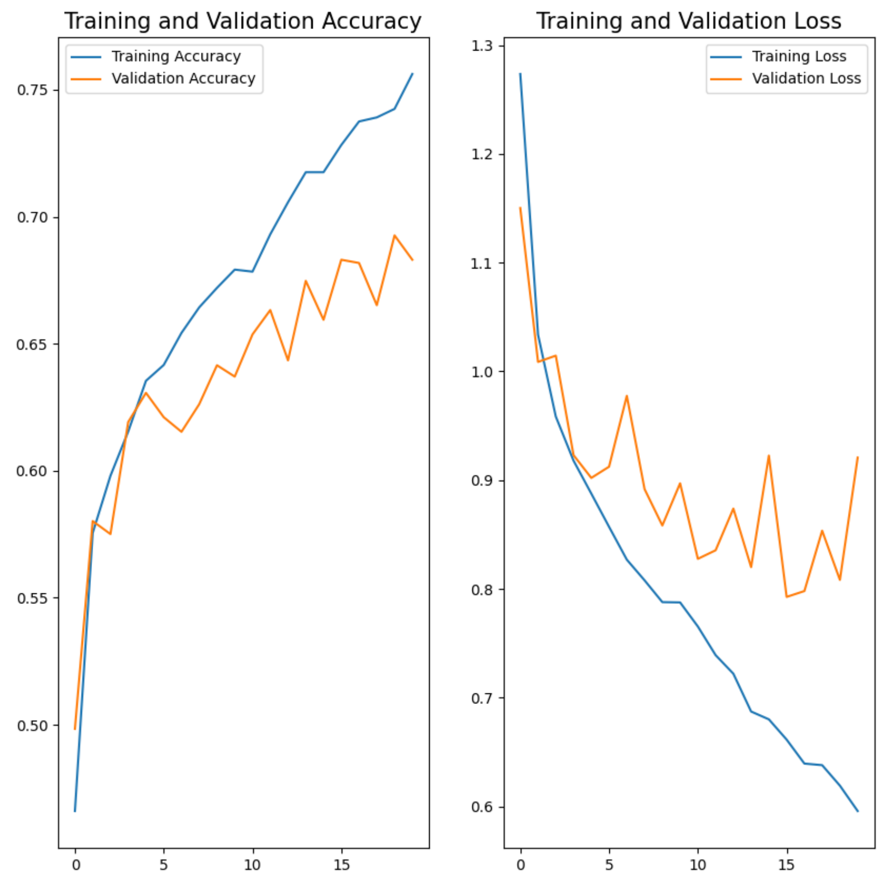
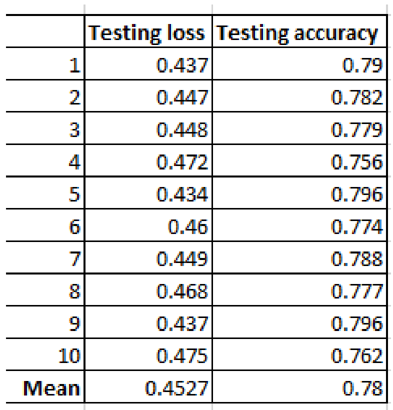


Fig. 15 – Accuracy and loss history during training of age model

5.2 Results of gender model

As the same dataset has been used as for age detection, the gender model also suffered from long training times when running 20+ epochs. Thus, the finalised model results are achieved with 20 epochs.



	Testing loss	Testing accuracy
1	0.437	0.79
2	0.447	0.782
3	0.448	0.779
4	0.472	0.756
5	0.434	0.796
6	0.46	0.774
7	0.449	0.788
8	0.468	0.777
9	0.437	0.796
10	0.475	0.762
Mean	0.4527	0.78

Fig. 16 – Gender model results

The finalised gender model achieved a loss of 0.453 and an accuracy of 78%, outperforming the age model. This could be due to the increased number of convolutional layers or the increased number of filters per layer, but as a result each epoch of the gender model took almost twice as long as the age model.

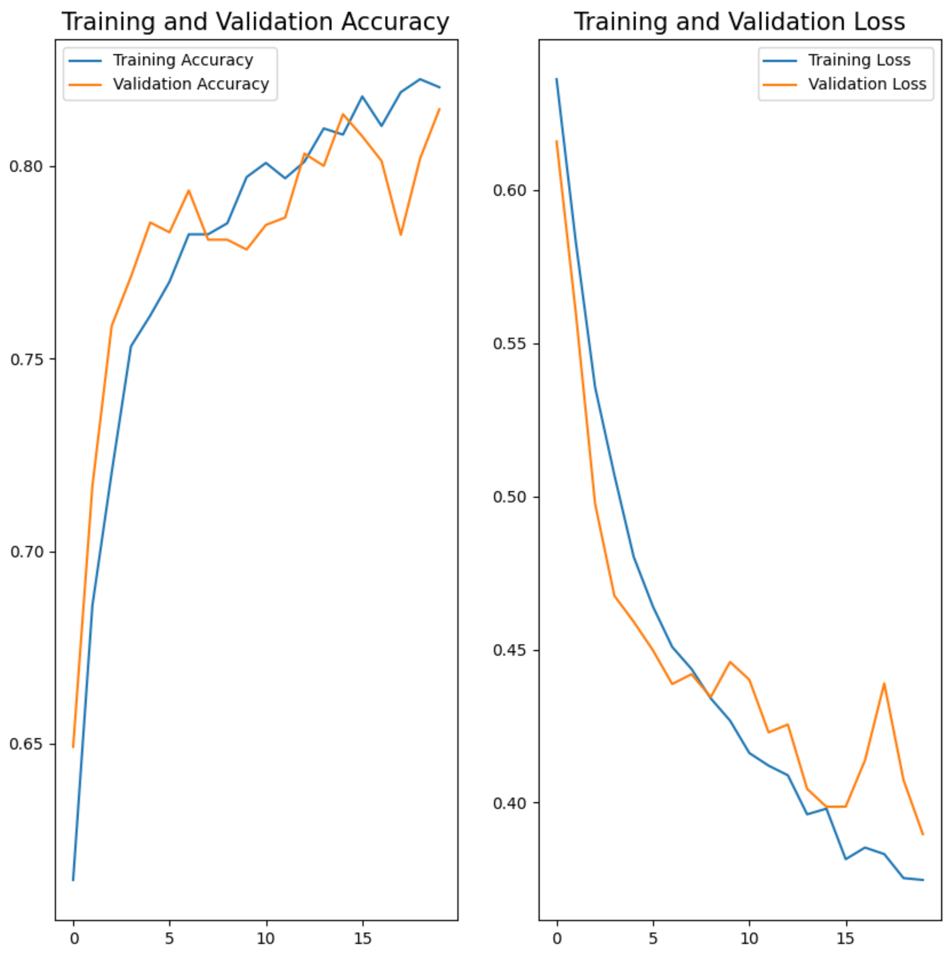
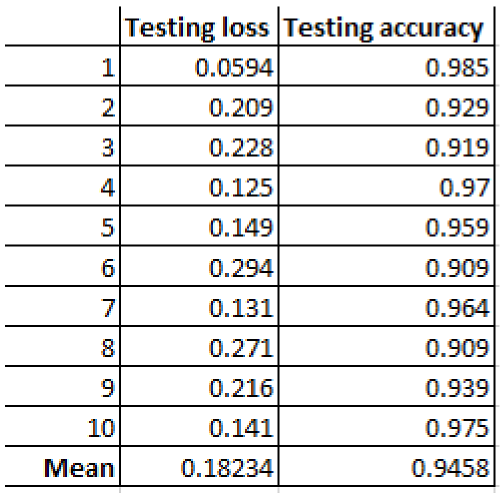


Fig. 17 – Accuracy and loss history during training of gender model

5.3 Results of facial expression model

The CK+48 dataset used for the facial expression recognition problem consists of ~900 images, a much smaller number than the ~20,000 used for age and gender recognition. As a result, despite being the deepest architecture of the three models, training the FER model is drastically shorter than training the age / gender models. Therefore, I have been able to repeatedly train and test the FER model with 50 epochs without any issue.



	Testing loss	Testing accuracy
1	0.0594	0.985
2	0.209	0.929
3	0.228	0.919
4	0.125	0.97
5	0.149	0.959
6	0.294	0.909
7	0.131	0.964
8	0.271	0.909
9	0.216	0.939
10	0.141	0.975
Mean	0.18234	0.9458

Fig. 18 – Facial expression model results

The finalised FER model achieved a mean loss of 0.182 and an accuracy of 94.6% - a very high result. The reasons for this model outperforming the age and gender models by a large margin likely stem from the increased number of parameters allowing for more complex feature extraction.

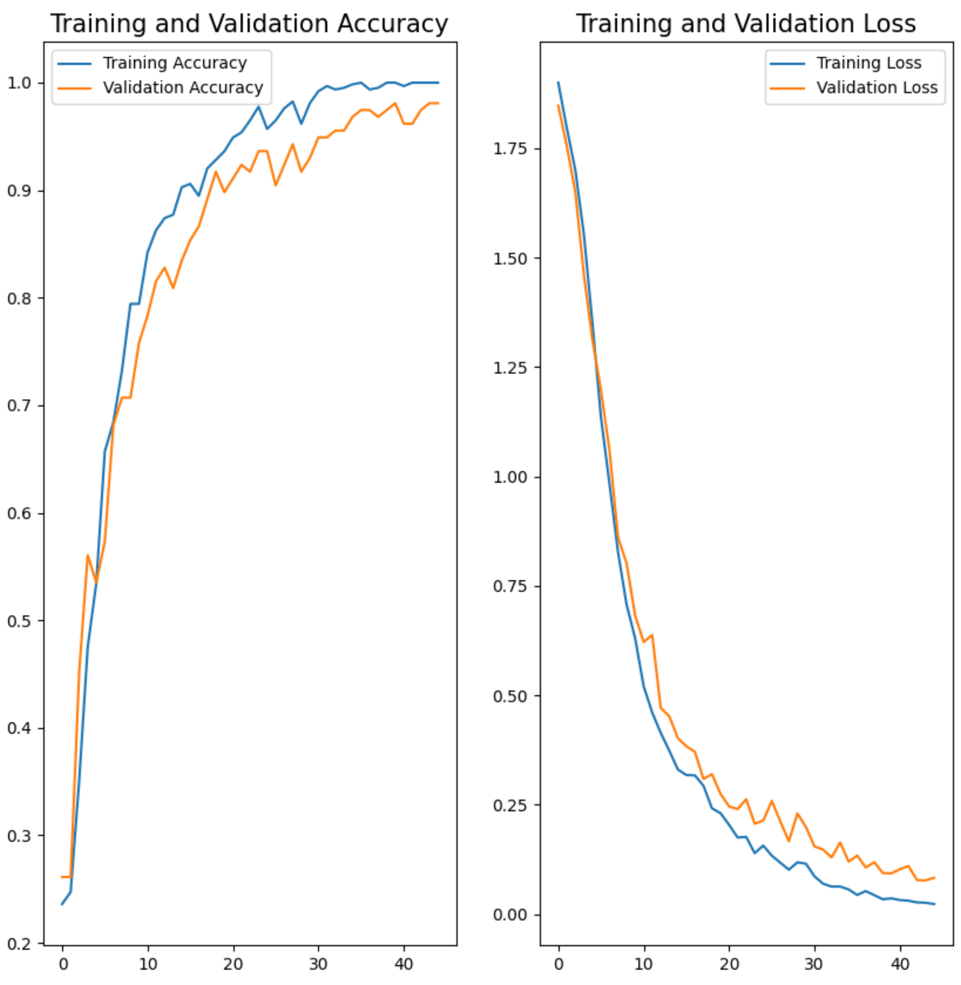
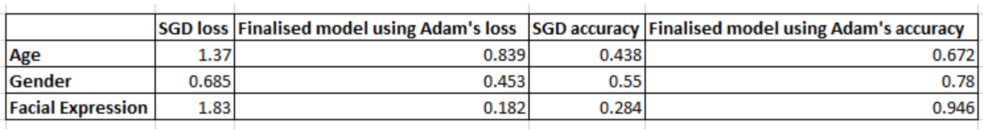


Fig. 19 - Accuracy and loss history during training of facial expression model

5.4 Results of model variations

In order to arrive at the finalised models described in Section 4, there was a need for extensive trial and error with different network architectures and hyperparameters.

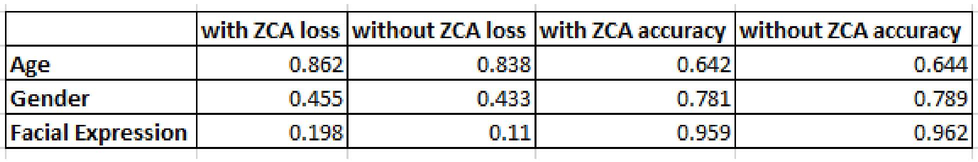
One of the properties I varied during my experiments was the choice of optimizer. Switching to SGD had a similar effect on all three models. The age, gender and facial expression models perform 23.4%, 23% and 66.2% worse respectively when using SGD than using Adam. Additionally, when observing the plots, it seemed clear that SGD was getting stuck at local minima – reinforcing the ideas from earlier research in Section 3.4 and demonstrating with empirical evidence the reason behind Adam optimizer’s current popularity in the field of deep learning. From these observations, it was clear that Adam would be the optimizer of choice for the finalised models of highest accuracies.



	SGD loss	Finalised model using Adam's loss	SGD accuracy	Finalised model using Adam's accuracy
Age	1.37	0.839	0.438	0.672
Gender	0.685	0.453	0.55	0.78
Facial Expression	1.83	0.182	0.284	0.946

Fig. 20 – Results of SGD vs Adam optimizer

For all three finalised models I have implemented horizontal flip and rotation (up to 45°) augmentations as both augmentations seemed to offer small increases in accuracy (~2-3% for age and gender models and up to ~5% for the facial expression models). However, I also choose to investigate the effects of Zero Component Analysis (ZCA) Whitening on the performance of each model:

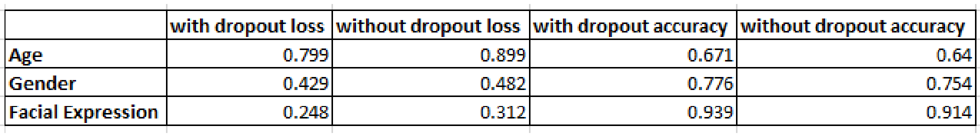


	with ZCA loss	without ZCA loss	with ZCA accuracy	without ZCA accuracy
Age	0.862	0.838	0.642	0.644
Gender	0.455	0.433	0.781	0.789
Facial Expression	0.198	0.11	0.959	0.962

Fig. 21 – Results of ZCA whitening vs No ZCA whitening

Interestingly, ZCA whitening had an almost negligible effect on the models' accuracies. However, the age, gender and facial expression models perform 0.2%, 0.8% and 0.3% worse respectively with ZCA than without ZCA. Though this is a small difference, I concluded that the highest overall accuracies would be reached without using the ZCA whitening augmentation. This is particularly surprising for the facial expression model, as I estimated that due to the small starting dataset, any additional augmentations would only increase the model's performance.

The training history plots for the age and gender detection models indicate that as training continues, the models may be overfitting to the training data. This is indicated by the training loss continuing to decrease, whilst the validation loss stagnates – at this stage, the ability for the network to generalise is no longer improving. In order to combat this, I experimented with alterations to the network architecture by adding Dropout() layers. Dropout is a regularisation method that randomly 'drops' some number of layer outputs. Often when training networks, layers can co-adapt to fix mistakes created by previous layers in the model, leading to overfitting as these co-adaptations are unable to generalise to unseen data [56]. Conceptually, dropout should stop such situations occurring, thus reducing overfitting.



	with dropout loss	without dropout loss	with dropout accuracy	without dropout accuracy
Age	0.799	0.899	0.671	0.64
Gender	0.429	0.482	0.776	0.754
Facial Expression	0.248	0.312	0.939	0.914

Fig.22 – Results of Dropout vs No Dropout

As expected, introducing dropout improved the performance of all three models by 3.1%, 2.2% and 2.5% respectively. Thus, all three of the finalised models include dropout layers that drop 20% of the input units after each convolutional layer.

6 Conclusion

In this project I have presented three models: for age, gender and facial expression recognition. The finalised models achieved best average accuracies of 67.2%, 78% and 94.6% respectively when evaluated on their testing sets. As the primary aim of this project was to achieve 90%+ accuracies, the age and gender models have failed to perform well enough to consider them highly successful. However, the facial expression model has met this aim and outperformed expectations by 4.6%. Extensive experiments were performed for all three models, in attempt to find their optimum configurations. One key limiting factor in this project has been the lack of sufficiently powerful hardware to train the age and gender models with a higher number of epochs. Due to the lack of GPU memory, it was unfeasible to run training cycles of 50+ epochs, which I believe would have led to a significant increase in the performances of both the age and gender models. Additionally, this GPU bottleneck made it impossible to experiment with input resolutions higher than 220 x 220, as there was simply not enough memory to allocate, and the models would not even begin to train. I was also restricted in the batch sizes that could be tested, as anything above 32 was taking impractically long to train.

The secondary aim of investigating the effects of certain variations on the age, gender and facial expression recognition models has been successful. I can conclude, with empirical evidence, that implementing Dropout and using Adam optimizer both have a positive impact on these models' testing losses and accuracies, whilst ZCA whitening (albeit very little) and SGD optimizer are detrimental. However, it is hard to say whether these conclusions will remain valid if one attempts to train the same models on different datasets.

Further research directions include investigating the potentially beneficial effect of BatchNormalization() layers as a regularisation technique to further reduce overfitting. Additionally, I did not have the chance to go as far as to experiment with Inception modules [40], which through allowing for more efficient computation during training, could have offered a temporary solution to the limitations in GPU memory.

Bibliography

[1] A. Dehghan, E. G. Ortiz, G. Shu, and S. Z. Masood, ‘DAGER: Deep Age, Gender and Emotion Recognition Using Convolutional Neural Network’, arXiv:1702.04280 [cs], Mar. 2017, Accessed: Mar. 11, 2021. [Online]. Available: http://arxiv.org/abs/1702.04280

[2] K. Liu, M. Zhang, and Z. Pan, ‘Facial Expression Recognition with CNN Ensemble’, in 2016 International Conference on Cyberworlds (CW), Chongqing, China, Sep. 2016, pp. 163–166. doi: 10.1109/CW.2016.34.

[3] R. Ranjan, S. Sankaranarayanan, C. D. Castillo, and R. Chellappa, ‘An All-In-One Convolutional Neural Network for Face Analysis’, arXiv:1611.00851 [cs], Nov. 2016, Accessed: Mar. 11, 2021. [Online]. Available: http://arxiv.org/abs/1611.00851

[4] G. Levi and T. Hassncer, ‘Age and gender classification using convolutional neural networks’, in 2015 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Boston, MA, USA, Jun. 2015, pp. 34–42. doi: 10.1109/CVPRW.2015.7301352.

[5] L. Wolf, T. Hassner, and Y. Taigman, ‘Descriptor Based Methods in the Wild’, France, Oct. 2008, p. 15.

[6] C. Shan, ‘Learning local binary patterns for gender classification on real-world face images’, Pattern Recognition Letters, vol. 33, no. 4, pp. 431–437, Mar. 2012, doi: 10.1016/j.patrec.2011.05.016.

[7] A. Greco, A. Saggese, M. Vento, and V. Vigilante, ‘A Convolutional Neural Network for Gender Recognition Optimizing the Accuracy/Speed Tradeoff’, IEEE Access, vol. 8, pp. 130771–130781, 2020, doi: 10.1109/ACCESS.2020.3008793.

[8] P. R. Dachapally, ‘Facial Emotion Detection Using Convolutional Neural Networks and Representational Autoencoder Units’, arXiv:1706.01509 [cs, stat], Jun. 2017, Accessed: Mar. 11, 2021. [Online]. Available: http://arxiv.org/abs/1706.01509

[9] N. Mehendale, ‘Facial emotion recognition using convolutional neural networks (FERC)’, SN Appl. Sci., vol. 2, no. 3, p. 446, Mar. 2020, doi: 10.1007/s42452-020-2234-1.

[10] A. Mollahosseini, D. Chan, and M. H. Mahoor, ‘Going deeper in facial expression recognition using deep neural networks’, in 2016 IEEE Winter Conference on Applications of Computer Vision (WACV),

Lake Placid, NY, USA, Mar. 2016, pp. 1–10. doi: 10.1109/WACV.2016.7477450.

[11] C. Pramerdorfer and M. Kampel, ‘Facial Expression Recognition using Convolutional Neural Networks: State of the Art’, arXiv:1612.02903 [cs], Dec. 2016, Accessed: Apr. 08, 2021. [Online]. Available: http://arxiv.org/abs/1612.02903

[12] Y. LeCun et al., ‘Backpropagation Applied to Handwritten Zip Code Recognition’, Neural Computation, vol. 1, no. 4, pp. 541–551, Dec. 1989, doi: 10.1162/neco.1989.1.4.541.

[13] K. Fukushima and S. Miyake, ‘Neocognitron: A Self-Organizing Neural Network Model for a Mechanism of Visual Pattern Recognition’, in Competition and Cooperation in Neural Nets, vol. 45, S. Amari and M. A. Arbib, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1982, pp. 267–285. doi: 10.1007/978-3-642-46466-9_18.

[14] J. Deng, W. Dong, R. Socher, L.-J. Li, Kai Li, and Li Fei-Fei, ‘ImageNet: A large-scale hierarchical image database’, in 2009 IEEE Conference on Computer Vision and Pattern Recognition, Miami, FL, Jun. 2009, pp. 248–255. doi: 10.1109/CVPR.2009.5206848.

[15] A. Krizhevsky, I. Sutskever, and G. E. Hinton, ‘ImageNet classification with deep convolutional neural networks’, Commun. ACM, vol. 60, no. 6, pp. 84–90, May 2017, doi: 10.1145/3065386.

[16] K. K. Pal and K. S. Sudeep, ‘Preprocessing for image classification by convolutional neural networks’, in 2016 IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT), Bangalore, India, May 2016, pp. 1778–1781. doi: 10.1109/RTEICT.2016.7808140.

[17] D. A. Pitaloka, A. Wulandari, T. Basaruddin, and D. Y. Liliana, ‘Enhancing CNN with Preprocessing Stage in Automatic Emotion Recognition’, Procedia Computer Science, vol. 116, pp. 523–529, 2017, doi: 10.1016/j.procs.2017.10.038.

[18] L. Perez and J. Wang, ‘The Effectiveness of Data Augmentation in Image Classification using Deep Learning’, arXiv:1712.04621 [cs], Dec. 2017, Accessed: Apr. 11, 2021. [Online]. Available: http://arxiv.org/abs/1712.04621

[19] C. Shorten and T. M. Khoshgoftaar, ‘A survey on Image Data Augmentation for Deep Learning’, J Big Data, vol. 6, no. 1, p. 60, Dec. 2019, doi: 10.1186/s40537-019-0197-0.

[20] K. He, X. Zhang, S. Ren, and J. Sun, ‘Deep Residual Learning for Image Recognition’, arXiv:1512.03385 [cs], Dec. 2015, Accessed: Apr. 11, 2021. [Online]. Available: http://arxiv.org/abs/1512.03385

[21] A. Krizhevsky, ‘Learning Multiple Layers of Features from Tiny Images’, p. 60, 2009.

[22] F. J. Moreno-Barea, F. Strazzera, J. M. Jerez, D. Urda, and L. Franco, ‘Forward Noise Adjustment Scheme for Data Augmentation’, in 2018 IEEE Symposium Series on Computational Intelligence (SSCI), Bangalore, India, Nov. 2018, pp. 728–734. doi: 10.1109/SSCI.2018.8628917.

[23] I. Goodfellow et al., ‘Generative Adversarial Nets’, p. 9, 2014.

[24] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, ‘Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks’, arXiv:1703.10593 [cs], Aug. 2020, Accessed: Apr. 21, 2021. [Online]. Available: http://arxiv.org/abs/1703.10593

[25] X. Zhu, Y. Liu, Z. Qin, and J. Li, ‘Data Augmentation in Emotion Classification Using Generative Adversarial Networks’, arXiv:1711.00648 [cs], Dec. 2017, Accessed: Apr. 21, 2021. [Online]. Available: http://arxiv.org/abs/1711.00648

[26] I. J. Goodfellow et al., ‘Challenges in Representation Learning: A Report on Three Machine Learning Contests’, in Neural Information Processing, vol. 8228, M. Lee, A. Hirose, Z.-G. Hou, and R. M. Kil, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 117–124. doi: 10.1007/978-3-642-42051-1_16.

[27] R. Rothe, R. Timofte, and L. V. Gool, ‘DEX: Deep EXpectation of Apparent Age from a Single Image’, in 2015 IEEE International Conference on Computer Vision Workshop (ICCVW), Santiago, Chile, Dec. 2015, pp. 252–257. doi: 10.1109/ICCVW.2015.41.

[28] Y. H. Kwon and N. da V. Lobo, ‘Age Classification from Facial Images’, Computer Vision and Image Understanding, vol. 74, no. 1, pp. 1–21, Apr. 1999, doi: 10.1006/cviu.1997.0549.

[29] N. Ramanathan and R. Chellappa, ‘Modeling Age Progression in Young Faces’, in 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 1 (CVPR’06), New York, NY, USA, 2006, vol. 1, pp. 387–394. doi: 10.1109/CVPR.2006.187.

[30] Y. Sun, X. Wang, and X. Tang, ‘Deep Learning Face Representation from Predicting 10,000 Classes’, in 2014 IEEE

Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, Jun. 2014, pp. 1891–1898. doi: 10.1109/CVPR.2014.244.

[31] E. Eidinger, R. Enbar, and T. Hassner, ‘Age and Gender Estimation of Unfiltered Faces’, IEEE Trans.Inform.Forensic Secur., vol. 9, no. 12, pp. 2170–2179, Dec. 2014, doi: 10.1109/TIFS.2014.2359646.

[32] M. Mathias, R. Benenson, M. Pedersoli, and L. Van Gool, ‘Face Detection without Bells and Whistles’, in Computer Vision – ECCV 2014, vol. 8692, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Cham: Springer International Publishing, 2014, pp. 720–735. doi: 10.1007/978-3-319-10593-2_47.

[33] B. Moghaddam and Ming-Hsuan Yang, ‘Learning gender with support faces’, IEEE Trans. Pattern Anal. Machine Intell., vol. 24, no. 5, pp. 707–711, May 2002, doi: 10.1109/34.1000244.

[34] C. BenAbdelkader and P. Griffin, ‘A Local Region-based Approach to Gender Classi.cation From Face Images’, in 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’05) - Workshops, San Diego, CA, USA, 2005, vol. 3, pp. 52–52. doi: 10.1109/CVPR.2005.388.

[35] G. B. Huang, M. Mattar, T. Berg, and E. Learned-Miller, ‘Labeled faces in the wild: A database for studying face recognition in unconstrained environments’, p. 15, 2008.

[36] A. G. Howard et al., ‘MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications’, arXiv:1704.04861 [cs], Apr. 2017, Accessed: Jun. 19, 2021. [Online]. Available: http://arxiv.org/abs/1704.04861

[37] C. Shan, S. Gong, and P. W. McOwan, ‘Facial expression recognition based on Local Binary Patterns: A comprehensive study’, Image and Vision Computing, vol. 27, no. 6, pp. 803–816, May 2009, doi: 10.1016/j.imavis.2008.08.005.

[38] S. M. Mavadati, M. H. Mahoor, K. Bartlett, P. Trinh, and J. F. Cohn, ‘DISFA: A Spontaneous Facial Action Intensity Database’, IEEE Trans. Affective Comput., vol. 4, no. 2, pp. 151–160, Apr. 2013, doi: 10.1109/T-AFFC.2013.4.

[39] ‘ImageNet Large Scale Visual Recognition Challenge 2012 (ILSVRC2012)’. https://image-net.org/challenges/LSVRC/2012/

[40] Y. Sun, D. Liang, X. Wang, and X. Tang, ‘DeepID3: Face Recognition with Very Deep Neural Networks’, arXiv:1502.00873 [cs],

Feb. 2015, Accessed: Jun. 21, 2021. [Online]. Available: http://arxiv.org/abs/1502.00873

[41] ‘Challenges in representation learning: Facial expression recognition challenge’. https://www.kaggle.com/c/challenges-in-representation-learning-facial-expression-recognition-challenge

[42] P. Lucey, J. F. Cohn, T. Kanade, J. Saragih, Z. Ambadar, and I. Matthews, ‘The Extended Cohn-Kanade Dataset (CK+): A complete dataset for action unit and emotion-specified expression’, in 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Workshops, San Francisco, CA, USA, Jun. 2010, pp. 94–101. doi: 10.1109/CVPRW.2010.5543262.

[43] ‘UTKFace: Large Scale Face Dataset’. https://susanqq.github.io/UTKFace/

[44] ‘TensorFlow’. https://www.tensorflow.org

[45] ‘Keras’. https://keras.io/

[46] ‘Keras Pre-processing’. https://keras.io/api/preprocessing/

[47] Z. Zhang, Y. Song, and H. Qi, ‘Age Progression/Regression by Conditional Adversarial Autoencoder’, arXiv:1702.08423 [cs], Mar. 2017, Accessed: Jun. 25, 2021. [Online]. Available: http://arxiv.org/abs/1702.08423

[48] G. E. Nasr, E. A. Badr, and C. Joun, ‘Cross Entropy Error Function in Neural Networks: Forecasting Gasoline Demand’, p. 4.

[49] D. P. Kingma and J. Ba, ‘Adam: A Method for Stochastic Optimization’, arXiv:1412.6980 [cs], Jan. 2017, Accessed: Jun. 26, 2021. [Online]. Available: http://arxiv.org/abs/1412.6980

[50] S. Ruder, ‘An overview of gradient descent optimization algorithms’, arXiv:1609.04747 [cs], Jun. 2017, Accessed: Jun. 26, 2021. [Online]. Available: http://arxiv.org/abs/1609.04747

[51] P. M. Radiuk, ‘Impact of Training Set Batch Size on the Performance of Convolutional Neural Networks for Diverse Datasets’, Information Technology and Management Science, vol. 20, no. 1, Jan. 2017, doi: 10.1515/itms-2017-0003.

[52]‘pandas.DataFrame’. https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.html

[53]‘train_test_split’. https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html

[54]‘ImageDataGenerator’. https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/image/ImageDataGenerator

[55] ‘Keras Callbacks’. https://keras.io/api/callbacks/

[56] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, ‘Dropout: A Simple Way to Prevent Neural Networks from Overfitting’, p. 30.

[57] ‘Facial Action Coding System’. https://www.paulekman.com/facial-action-coding-system/

[58] ‘Keras Accuracy Class’. https://keras.io/api/metrics/accuracy_metrics/#accuracy-class