Computer vision on images that predicts age, gender and emotions

Joakim Kvistholm

NBI/Handelsakademin

Master Thesis - Artificial Intelligence

2024-12

Abstract

In this report machine learning is used to identify emotions, the age (given in twenty-year categories), and the gender from images. The images for the emotions are obtained from a special dataset from Kaggle that contains seven different emotions on which the models are trained and validated. The age and gender classifications, on the other hand, are based on a lot of combined datasets, primarily from Kaggle. To solve the problem of identifying emotions, the age and the gender in an image, computer vision is used in python with neural networks, which contain convolutional layers and dense hidden layers. I used packages such as numpy, pandas, matplotlib, tensorflow and keras in scikit learn. Once the model is trained by a neural network, it is used in a real time application. This is another python program that loads the neural network model and then starts a laptop web camera that scans a face in real time and then displays the emotion, the age and the gender of the scanned face. To improve the performance of the neural network, when it is trained, I used l2 regulators and chose the learning rate carefully with small values. I noticed that it is better to train the neural networks slowly.

One of the goals of the project is to get an accuracy of at least 0.65 on the prediction of an emotion in an image. The goal of this accuracy score is achieved, but only by a small margin. Other goals are that the correct age category is obtained with an accuracy of at least 0.70 and that the correct gender is obtained with an accuracy of at least 0.95. Both accuracy scores are achieved, but again only by a small margin.

Contents

1 Introduction

The goal of this project is to use computer vision to identify different emotions, the age (category) and the gender from images. To achieve this goal machine learning with neural networks is used. The neural networks for image analysis contain convolutional layers and dense hidden layers. Once the architecture of the neural networks is decided, they are then trained on a lot of images with different emotions, age or gender and they are then saved for further use. The saved models are then used in another python program to identify the emotions, the age and the gender in the images, which are obtained from a laptop's web camera in real time.

The images that are used for emotion detection are of the size 48 x 48 pixels and are grayscale images (with a one-color channel). The images for training and validation are from Kaggle and the dataset is called "Facial Recognition Dataset" (that can be obtained from the following link: **https://www.kaggle.com/datasets/apollo2506/facial-recognition-dataset**). Some problems with the images in dataset are that the resolution is low, the contrast is low and the number of images of some of the emotions, as "disgust", are few. In the data set there are seven emotions: "angry", "disgust", "fear", "happy", "neutral", "sad" and "surprise".

The images that are used for the "age categories" and the "gender" detection are of the size 48 x 48 pixels and are color images (with three color channels, RGB or RGBA). The images for training and validation are primarily obtained from Kaggle and many of the images are unfortunately of poor quality. Some of the images are also mislabeled or contain more than one face. The images for age and gender detection are combinations of the following datasets:

1. "Facial age" (D1): https://www.kaggle.com/datasets/frabbisw/facial-age
2. "Faces: Age Detection from Images" (D2): https://www.kaggle.com/datasets/arashnic/faces-age-detection-dataset
3. "Gender Detection & Classification - Face Dataset" (D3): https://www.kaggle.com/datasets/trainingdatapro/gender-detection-and-classification-image-dataset
4. "Male and female faces dataset" (D4): https://www.kaggle.com/datasets/ashwingupta3012/male-and-female-faces-dataset
5. "Age Detection Object Detection Dataset by FluxBlazeSU2022" (D5): https://universe.roboflow.com/fluxblazesu2022/age-detection-tom31
6. "Gender Classification Dataset" (D6): https://www.kaggle.com/datasets/cashutosh/gender-classification-dataset
7. "In-the-wild Faces" (D7): https://susanqq.github.io/UTKFace

For this project the chosen programming environment is python with packages from scikit learn as numpy, pandas, matplotlib, keras and tensorflow. These packages contain the neural network environment (especially keras) with suitable mathematical tools as tensorflow.

This work is in some part a continuation of Akshit Madan's code and work on the YouTube video given by the following link: **https://www.youtube.com/watch?v=Bb4Wvl57LIk**. The code in the video is an old code that contains deprecated functions and imports for a lot earlier versions of python, keras and tensorflow and can't directly be used without changing "a lot of the code" (or at least some important parts of the code).

This technique, which is used in this project, to identify emotions, the age and the gender from images or cameras could be used by salesmen, guards and in warehouses to find suitable customers. A similar technique could be used to identify other data as clothing, hair colour or ethnicity from a different set of images. This could be used for dating services or criminal analysis. There are of course more areas where this or similar techniques could be used as recognizing workers in a factory or getting statistical data about workers in different departments of the factory (if they are using cameras in the factory).

The main goal of this project is to create an AI model that can predict human emotions, the age (category) and the gender from images, see fig 1.1. The goal of this project is twofold. The "first" goal is to create working programs in python and the "second" goal is to achieve a chosen accuracy score. The following goals formulated as tasks and questions will be answered in this thesis:

1. Can I create programs in python that train neural networks to identify the emotion, the age category and the gender from the images in the dataset with models that can be saved?
2. Can I create a program in python that loads trained models and then uses these models to identify the emotion, the age category and the gender from the images, which are obtained from a laptop's web camera in real time?
3. Can I achieve an accuracy score of at least 0.65 from one of my trained models when detecting "emotions"?
4. Can I achieve an accuracy score of at least 0.70 from one of my trained models when detecting "age categories"?
5. Can I achieve an accuracy score of at least 0.95 from one of my trained models when detecting "gender"?

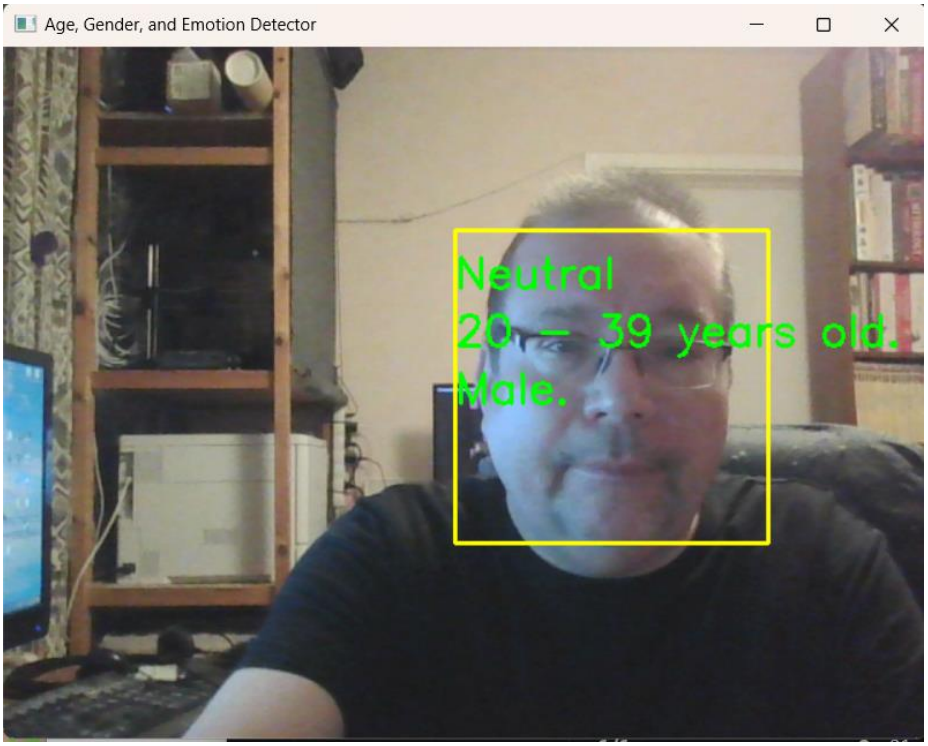

Figure 1.1. An illustration of the main program, "main.py", is shown in the image. The laptop's web camera captures images to detect emotions, the age category and gender, and then displays these results in the captured image.

2 Theory

2.1 Neural networks

The initial idea behind neural networks comes from a simplified neurological model of the human brain. The brain contains a network of neurons. One neuron receives a lot of signals from other neurons. If the summed signals that reach the neuron are larger than a certain threshold value, it sends signals to other neurons (otherwise it won't send signals). This threshold functionality corresponds to an "activation" function in a neural network architecture.

In the neural networks that are used in machine learning, the model learns a pattern or a relation between some input and output data. It finds a "black box function" for the dataset, that it has been trained for. A neural network has an input layer, an output layer and at least one or more hidden layers. The extra layers are hidden layers and can, for instance, be dense layers or convolutional layers. It is common to have many layers in a neural network. The fully connected layers, that are commonly used in neural networks, are called dense layers (see fig 2.1 and fig 2.2).

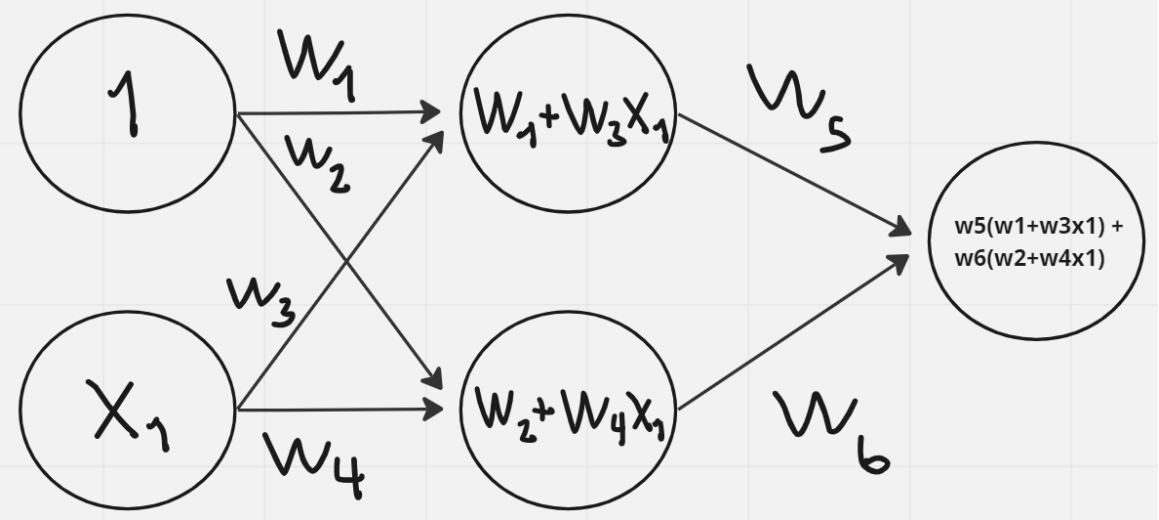

Figure 2.1. The architecture and the needed calculations for a simple neural network are shown. The extra node in the first layer with the value one in it adds a bias term to all nodes in the next layer (taken from [1]).

The main architecture of a neural network with *dense layers* is that it has layers that are fully connected, and that each layer contains a specific number of neurons, that can differ between layers. Each neuron in one layer is connected to all the neurons in the next layer. In a neuron in the next layer the node values from the preceding layer are multiplied with corresponding weights connected to this neuron and are then summed (see fig 2.1 and fig 2.2). On this summed value a function can be used, it is then called an "activation function". To each neuron in the preceding layer a bias term is normally added to get better flexibility for which type of patterns the model can learn. If no activation function is used the neural network will work as a "pure" regressor (with prediction). It is therefore common that activation functions are used in most neural networks, so that they can learn more complex patterns (and not only "linear patterns"). The function "relu" is very often used for this purpose (a straight line ($y = x$, $x > 0$) for positive values, or zero ($y = 0$, $x < 0$) for negative values).

Other common activation functions (mainly used in the output layer) are “softmax” for multiple classifications and “sigmoid” for binary classifications.

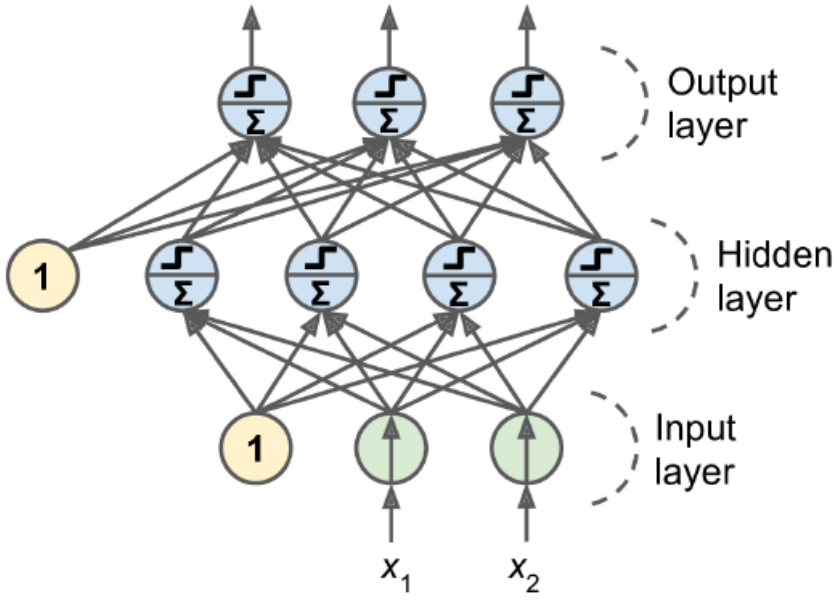

Figure 2.2. The architecture for a simple neural network with an input layer, an output layer and a hidden dense layer is shown. The extra node in each layer with the value one in it adds a bias term to all nodes in the next layer (taken from [1]).

To get a “prediction” (an output) from an input data a “forward propagation” is used. The calculations start from the input data and traverses through the different layers (in the neural network) all the way to the output data (see fig 2.1 and fig 2.2).

When a neural network is trained the weights and biases are updated. The steepest descent method or the gradient descent method (in mathematics) is used to minimize an error function (connected to the squared error between the predicted and the correct output value). In this case a backward propagation algorithm is used. The neural network is traversed in the reverse order, from the output data then through the layers and finally to the input data. The reason for this is that the gradient of the error (closely related to the output) needs to be calculated and for this purpose the chain rule is used. The chain rule goes from “the outer to the inner function” why the neural network will be traversed backwards and hence the name “backward propagation”. The squared error is differentiated with respect to the weights or the biasis. The direction of the gradient is the direction the function increases the most. If the opposite direction of the gradient is used, the function decreases the most and hence a minus sign is used when calculating the gradient. To scale how much the neural network should be trained, the gradient is multiplied by a factor, the learning rate. If the learning rate is too large the training can be unstable and if the learning rate is too low the training can be too slow. Once the gradient descent is calculated and scaled with the learning rate, the weights and biases are then updated from these calculations.

The reason some solutions differ or become larger than other solutions, can be because the function to be learned can contain one or more local minima (except the global minimum). The more complex the neural networks get, with more hidden layers and more nodes in each layer (that uses activation functions in the different layers) the more complex patterns it can find, but the training will be slower, and the risk of overfitting will be larger. (See [1] and [2].)

2.2 Convolutional neural networks

Convolutional neural networks are often used in image analysis and computer vision. Neural networks that contain convolutional layers are called convolutional neural networks (CNN).

The convolutional layers have a filtering function that splits the image into many small images, that are analyzed and used, to extract information of the whole image. The filtering function consists of two parts, feature maps and kernels (that for instance are of the size of 2 x 2). The kernel can be described as a window that moves across the image data where different smaller parts of the image are studied. The feature map is a kind of weight matrix (filter) with the same size as the kernel, that can be multiplied with a corresponding matrix of the same size, that is retrieved from a small part of the original image data. This "window" can be traversed across the entire image. The small images can contain information such as how the mouth is curved or if the eyes are open or closed. The information from all the small and "filtered" images can together get information about the whole image, which in turn can identify, for instance, the emotion that is displayed in the image. The neural network "chooses" the architecture of the filtering function and what each window represents by itself. A max pooling layer is often connected to a convolutional layer. (See [1] and [2].)

2.3 Other important concepts in neural networks

A *max pooling layer* (with the function "MaxPooling2D" in python) with, for instance, a kernel size of 2 x 2 can also be added to the convolutional layer. This layer also contains a filtering function that splits the image into small images of size 2 x 2. In each 2 x 2 size image data, the largest value is obtained, and this maximum value is then stored in a new matrix, that contains an important part of the original image data. This new matrix (of size 24 x 24) is reduced in size compared to the original image data matrix (of size 48 x 48) and contains more "compact" information.

Batch normalization can also be added to a convolutional layer. It is a regularizing method. It rescales the values to or from nodes in a layer (depending on implementation). The values are standardized with a mean of zero and a standard deviation of one. This avoids saturations during the training. This means that the training will be more stable, and the training will be less overfitted. A larger value on the learning rate can many times be used, when this regularization is used, without compromising the quality of the training.

Yet another important regularization method is *dropout* that is connected to a certain layer in the neural network. It has a parameter that corresponds to a probability p, that can be, for instance, 0.25. This means that there is a 25 % chance that a certain node in a layer will not be used (this is checked for all the nodes in the chosen layer). This also has a stabilizing effect on the training of a neural network. This type of regularization can also be added to convolutional layers.

Another good regularization method that is often used in neural networks is *l2 regularization*. It is based on "Ridge Regression like method" from regression analysis theory. Certain weights and therefore nodes will be set to zero (or close to zero). A parameter is set to choose how strong this effect will be. A larger value on this parameter gives a more restricting setting on the training. Other neurons (that are not set to zero) in the layer with the regularization need to take over the training of the model. This regularization method also has a stabilizing effect on the training of a neural network. This type of regularization can be added to convolutional layers, but also to dense layers.

Another good method to improve a neural network model is to use *transfer learning*. A pre-trained model that solves a similar problem (to our problem) could be used, to achieve better and faster training of the neural network model. A lot of important layers, especially the convolutional layers, are "saved" or "retrieved" from the pre-trained model to our own model, before we are training it. Many of the pre-trained layers can be frozen (and not changed) during the initial training to better keep the pre-trained model's "knowledge" to solve problems. This method can create a model with a good performance (accuracy) although we didn't have "enough" training data.

If the input data is too "clean" or "perfect", small random deviations can be introduced to improve the training and the performance of the neural network, when used on input data that contains small random deviations (which happens in many normal circumstances).

In this project *deep learning* (DL) is used. This means that the neural networks, that are trained, contain many hidden layers (with activation functions), so that the model can solve more complex non-linear problems such as image analysis or computer vision. (See [1], [2], [3], [4], [5] and [6].)

2.4 Accuracy score and confusion matrix

There are two metrics that are often used when checking the performance of a model (in neural networks), one is loss, and the other one is accuracy. The *accuracy score* is given by:

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad \text{... (1)}$$

Where TP is True Positive, TN is True Negative, FP is False Positive and FN is False Negative. The accuracy score is the number of correct predicted images divided by the total number of images in the test, see formula (1).

A *confusion matrix* is a good way to visualize the model's performance by looking at the predicted classes and comparing them with the actual classes. It shows the type of errors the model makes when predicting (and in this case classifying) the dataset. A simple representation of a confusion matrix can be seen below, see table 2.1. The correct predictions are in the main diagonal in the matrix. The false negative predictions are in the right upper part of the matrix and the false positive predictions are in the left lower part of the matrix. (See [1] and [2].)

	Predicted Positive	Predicted Negative
Actual Positive	TP	FN
Actual Negative	FP	TN

Table 2.1. The main structure of a confusion matrix is shown in the table.

3 Method

To solve the problem of identifying different emotions, age and gender from images machine learning is used. In this project computer vision is used in python with neural networks, which contain convolutional layers and dense hidden layers. The convolutional layers are needed because they are good at identifying different information of images such as “emotions”. To be more precise the neural network identifies many small aspects of the image as how the mouth is curved or if the eyes are open or closed which in turn can identify the emotion that is displayed in the image.

The neural networks are implemented in python with packages as numpy, pandas, matplotlib, tensorflow and keras in scikit learn. Especially keras and tensorflow are important to simplify the work with neural networks (to be used in a “Jupiter notebook”). In the first three python programs the AI models are trained and saved, and then the trained AI models are loaded and used in another python program. The other python program then uses these trained AI models to identify emotions, age and gender in images, which are obtained from a laptop’s web camera in real time.

The data, in this case the images for detecting emotions, are obtained from a special dataset, called “Facial Recognition Dataset”, from Kaggle that contains seven different emotions on which the models are trained and validated. The images that are used are of size 48 x 48 pixels and are grey scale images (with a one-color channel). In the dataset there are seven emotions: “angry”, “disgust”, “fear”, “happy”, “neutral”, “sad” and “surprise”.

The images are loaded from Kaggle and stored on a local hard drive (in folders). It is possible to use a “generator function” in python to retrieve the images but I loaded the images directly from the hard drive into the program’s memory as “x” (image information) and “y” (emotion label) variables. On the “y” variables (the emotion labels) I used one-hot encoding so that this data could be used to train the neural networks. In this project I used the above mentioned method of loading the data into memory from the hard drive so that I could have a direct access to the “x” and “y” variables when I needed to calculate the accuracy score and the confusion matrix for the trained neural network model. This method also makes the training of the neural networks faster. A similar process is used to obtain the images and the labelling of the categories or classes that are used for detecting age and gender from the images. In the case of age and gender detection, colour images of size 48 x 48 pixels with three color channels are used.

The main setting in the training of the neural networks that detect emotions is to use 48 or 50 epochs. Each epoch contains a special number of training steps. Four convolutional layers are used with Batch Normalization, Dropout (with a dropout rate of 0.25) and MaxPooling2D (with pooling size of (2 x 2)). The kernel sizes are (3 x 3) or (5 x 5). The chosen sizes of the four convolutional layers are 64, 128, 512 and 512. Two to three dense hidden layers are used. There is an input layer with the shape (48, 48, 1) and an output layer with 7 outputs. The output layer uses “softmax” as activation function and all other layers use “relu” as activation function. An l2 regularization is used in all layers or not at all. The compiler “Adam” is used with the loss function “categorical_crossentropy”. The image data is divided by 255 so that the greyscale “RGB” values lies between 0 and 1 (instead of between 0 and 255). This is done both for the image data in the generator set and for the image data that is directly retrieved from the hard drive (and then stored as variables in the python program).

The following variations are used during the training of the neural networks for *emotion* detection:

- Two to three dense hidden layers are used. Each layer has values from 256 to 1024 nodes (“neurons”).

- The learning rate is often close to 0.001. It also has larger and smaller values than this.
- The l2 regularization parameter is often set to 0.01. Larger and smaller values are also used. Also tried without l2 regularization.
- Tried to change the kernel sizes of the two last convolutional layers to (4 x 4) instead of (3 x 3).
- An early stopping is used with different patience rates (with values as 5, 7 and 10).
- A "reduced learning rate", with the function "ReduceLROnPlateau", is used (with different values on the "scaling parameter" but it is often set to 0.5). The patience rate is set to 3.

In this project over 21 different settings are used to create different models, on the neural networks to be trained ({sim1 (model 1), …, sim21 (model 21)}), to find a good model that could predict emotions from images. The same naming convention is also used to find good models to predict age and gender from images. Over 36 different settings on the neural networks are used to predict age and over 26 different settings are used to predict gender.

The main settings to detect age (categories) and gender are almost identical with the one that detects emotions. One difference is that instead of using greyscale images as input data, color images are used. The number of epochs is set to 50. The compiler "Adam" is mostly used, but other compilers are also used, all with the loss function "categorical_crossentropy". An early stopping is used with a patience rate of 10 and a "reduced learning rate" is used with the "scaling parameter" set to 0.5 and the patience rate set to 3. The image data for age and gender detection is also divided by 255, just as is done for the images that are used to detect emotions.

The following variations are used during the training of the neural networks for *age* detection:

- Two to three dense hidden layers are used. Each layer has values from 256 to 4024 nodes ("neurons").
- The learning rate is often close to 0.0005. It also has larger and smaller values than this.
- The l2 regularization parameter is often set to 0.01. Larger and smaller values are also used.
- The optimizer Adam is mostly used, but other optimizers such as RMSprop, Nadam and SGD are also used.

The following variations are used during the training of the neural networks for *gender* detection:

- Two to three dense hidden layers are used. Each layer has values from 1024 to 4024 nodes ("neurons").
- The learning rate is often close to 0.0005. It also has larger and smaller values than this.
- The l2 regularization parameter is often set to 0.01. Larger and smaller values are also used.
- The optimizer Adam is mostly used, but other optimizers such as RMSprop, Nadam and SGD are also used.

The composition for how the training and validation data is spread among the different categories for emotion, age and gender detection can be seen in table 3.1, table 3.2 and table 3.3.

Emotion	Angry	Disgust	Fear	Happy	Neutral	Sad	Suprise
Training data	3993	436	4103	7164	4982	4938	3205
Validation data	960	111	1018	1825	1216	1139	797

Table 3.1. The composition of how the training and validation data is spread among the different *emotions* can be seen in the table. The training data consists of 28821 images and the validation data consists of 7066 images. The entire dataset consists of 35887 images.

Age (category)	“000”	“020”	“040”	“060”	“080”
Training data	8153	12112	5294	2499	925
Validation data	2155	3022	1265	584	199

Table 3.2. The composition of how the training and validation data is spread among the different *age categories* can be seen in the table. The notation “000” means individuals between 0 and 19 years of age and “080” means individuals that are 80 years of age or older. The training data consists of 28983 images and the validation data consists of 7225 images. The entire dataset consists of 36208 images.

Gender	Female	Male
Training data	27430	28887
Validation data	6753	8398

Table 3.3. The composition of how the training and validation data is spread among the different *genders* can be seen in the table. The training data consists of 56317 images and the validation data consists of 15151 images. The entire dataset consists of 71468 images.

Gender detection is typically a *binary classification* problem but is treated and generalized as a *multiple classification* problem in this project. The reason behind this is that it maintains a consistent structure in the code and in the training of the neural networks. To achieve this goal the neural networks, that detect gender, are using a “*softmax*” activation function in the output layer with *two* output neurons, which corresponds to a multiple classification problem, instead of using a “*sigmoid*” activation function with *one* output neuron, that is typically used in a binary classification problem.

4 Results and Discussion

In this part the results from our trained neural networks are presented. Machine learning with neural networks is used to identify emotions, age and gender in images. This means that, for instance, our neural networks for emotion detection need to classify the images into seven different emotions (classes). The ages are classified in five "age categories" (in twenty-year intervals) and the gender is classified in two classes (female or male).

Only the best trained models are studied more closely. A model's performance is decided by an accuracy score on the validation data (where all the validation data is used). This accuracy score is calculated both for the model's last save during training and from the model's best epoch during training. In this project over 21 models are trained for detecting emotions. Another way to study how well a model is trained is to use a confusion matrix. The training history of the accuracy score and the losses for both the training and the validation data show if the trained model is overfitted, see fig 4.1, fig 4.3 and fig 4.5.

4.1 Results from the three best models for emotion detection

Model	Important settings (for the training of the model)	Accuracy (from the last saved model)	Accuracy (from the best epoch)
Model 6	Layers: hdl1(1024), hdl2(512). lr = 0.001, l2(0.01). EarlyStopping(patience = 5). ReduceLROnPlateau(factor = 0.5).	0.6696416	0.6696416
Model 9	Layers: hdl1(512), hdl2(512). lr = 0.001, l2(0.01). EarlyStopping(patience = 5). ReduceLROnPlateau(factor = 0.5).	0.6680833	0.6680833
Model 14	Layers: hdl1(1024), hdl2(512). lr = 0.0005, l2(0.01). EarlyStopping(patience = 10). ReduceLROnPlateau(factor = 0.5).	0.6754498	0.6754498

Table 4.1. The three best models for emotion detection are shown in the table. The callbacks that are used are early stopping and "ReduceLROnPlateau" with the "scale factor" that the learning rate, lr, is multiplied with (patience is here 3). An l2 regularization is used in all hidden layers with the "strength" given by the value in parenthesis. The notation "hdl1" means that it is the first hidden dense layer with the number of neurons that is given by the value inside the parenthesis.

The three best models for detecting *emotions* and their accuracies can be seen in table 4.1. The best model to detect emotions has an accuracy score of 0.67545 which is better than our goal in this project to get an accuracy score of 0.65. All models in the table have an accuracy score that exceeds the goal of 0.65. A lot of the trained models also have an accuracy score above 0.65. The accuracy scores for the trained models in this project are between 60 % and 68 % for detecting emotions. Most of the models are consistent in the sense that they are stable and gave similar results when they predicted emotions from images.

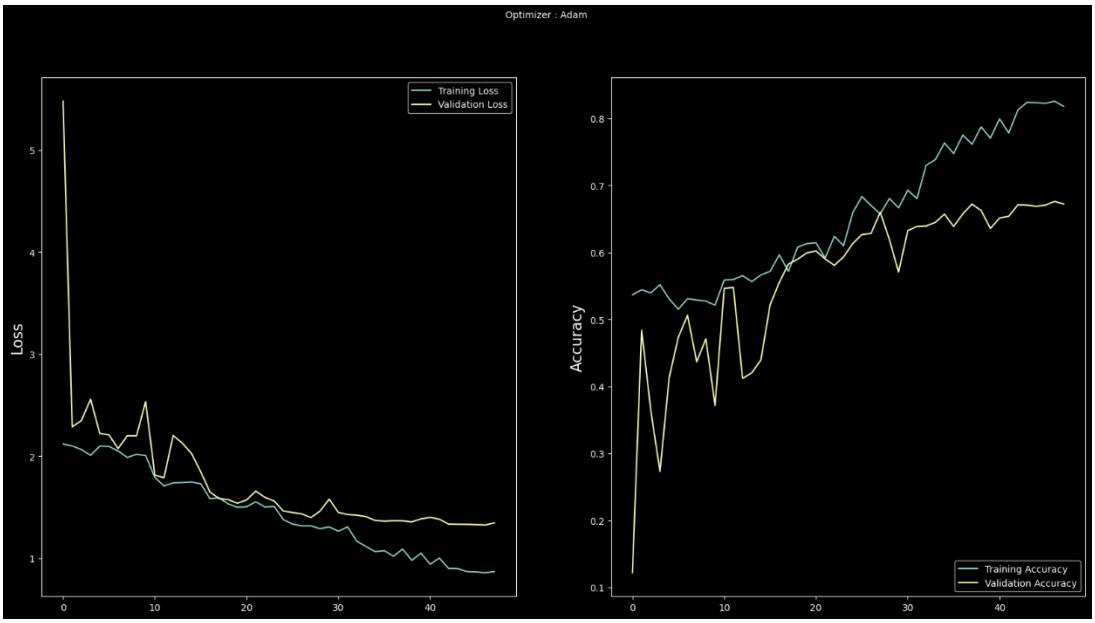

Figure 4.1. The history of the loss and the accuracy during training for the best model (model 14) for emotion detection is shown in the graph. The yellow line is the loss or accuracy on the validation data and the blue line is the loss or accuracy on the training data.

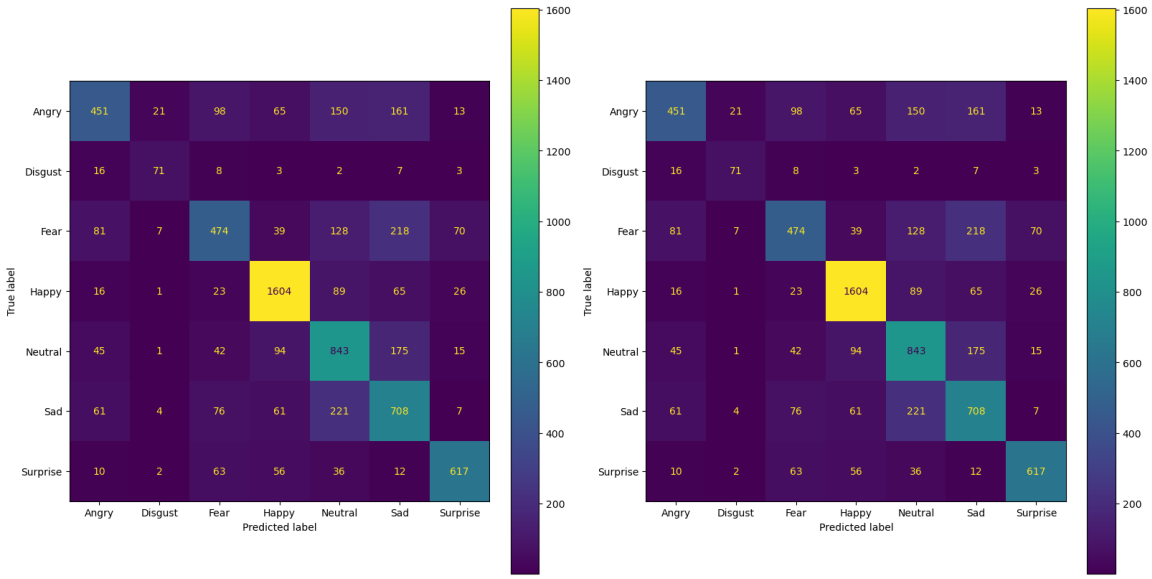

Figure 4.2. The confusion matrices for the best model (model 14) for emotion detection are shown in the graph. On the right the confusion matrix for model 14 for the last saved model can be seen and on the left the confusion matrix for model 14 for the best epoch can be seen.

The most common variation between the simulations are two or three dense layers, the number of neurons in each hidden dense layer, the learning rate, the l2 regularization and the settings for the callback functions as “EarlyStopping” and “ReduceLROnPlateau”. Slow training with a low learning rate and many epochs gave the best results. Using three layers didn’t give a better result when detecting emotions but having more neurons in the first layer gave a little better result. For all the trained models some level of overfitting is seen, see the graphs for model 14 in fig 4.1 as an illustration of this effect.

The confusion matrix in fig 4.2 illustrates the fact that some combinations of emotions are hard to separate into the right categories. It is difficult to see the difference between emotions as “sad and neutral”, “sad and fear”, and “neutral and fear”. One reason for this could be that the labeling of the emotions for the images is not entirely correct and another reason may be that it is difficult to make distinction between certain types of emotions.

4.2 Results from the three best models for age detection

Model	Important settings (for the training of the model)	Accuracy (from the last saved model)	Accuracy (from the best epoch)
Model 8	Layers: hdl1(1024), hdl2(1024). lr = 0.001, l2(0.01), opt = Adam. EarlyStopping(patience = 10). ReduceLROnPlateau(factor = 0.5).	0.7342561	0.7338408
Model 11	Layers: hdl1(1024), hdl2(1024), hdl3(2024). lr = 0.0005, l2(0.01), opt = Adam. EarlyStopping(patience = 10). ReduceLROnPlateau(factor = 0.5).	0.7245675	0.7381315
Model 19	Layers: hdl1(1024), hdl2(1024), hdl3(3024). lr = 0.0005, l2(0.01), opt = Adam. EarlyStopping(patience = 10). ReduceLROnPlateau(factor = 0.5).	0.7253979	0.7348097

Table 4.2. The three best models for age detection are shown in the table. The callbacks that are used are early stopping and “ReduceLROnPlateau” with the “scale factor” that the learning rate, lr, is multiplied with (patience is here 3). An l2 regularization is used in all hidden layers with the “strength” given by the value in parenthesis. The notation “hdl1” means that it is the first hidden dense layer with the number of neurons that is given by the value inside the parenthesis.

The three best models for detecting the *age category* and their accuracies can be seen in table 4.2. The best model to detect the age category has an accuracy score of 0.73813 which is better than our goal in this project of an accuracy score of 0.70. All models in the table have an accuracy score that exceeds the goal of 0.70. A lot of the trained models also have an accuracy above 0.70. Most of the

models are consistent in the sense that they are stable and gave similar results when they predicted the age category from images.

Slow training with a low learning rate and many epochs gave the best results. For all the trained models to detect the age category some level of overfitting is seen, see the graphs for model 11 in fig 4.3 as an illustration of this effect.

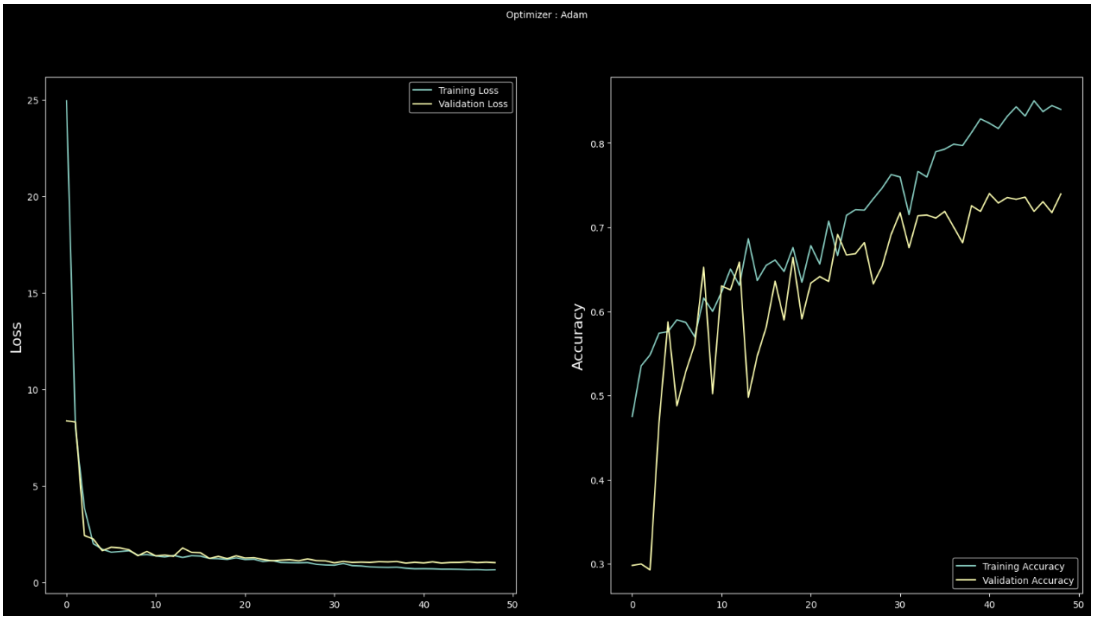

Figure 4.3. The history of the loss and the accuracy during training for the best model (model 11) for age detection is shown in the graph. The yellow line is the loss or accuracy on the validation data and the blue line is the loss or accuracy on the training data.

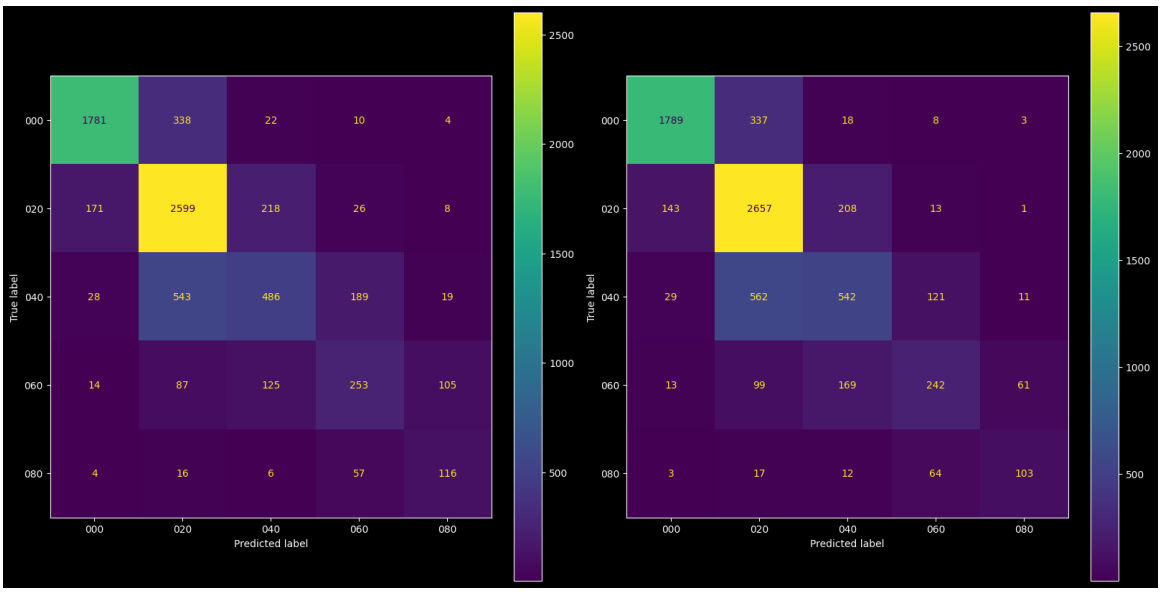

Figure 4.4. The confusion matrices for the best model (model 11) for age detection are shown in the graph. On the right the confusion matrix for model 11 for the last saved model can be seen and on the left the confusion matrix for model 11 for the best epoch can be seen.

The confusion matrix in fig 4.4 illustrates the fact that some ages are hard to separate into the right categories. The age categories have some natural ordering between them, and it is more difficult to separate two age categories from each other, if the ages lie close to each other, such as, for instance, individuals that are 19 and 20 years old. It is also difficult to find the right age category for very old individuals, since there isn't so much training data for the last age category "080" (80 years old or older). Another reason that it can be difficult to find the right age category could be that the individual's actual age in many cases doesn't have a simple one-to-one correspondence with the individual's appearance. The number of images in each age category is very unevenly distributed between the age categories. It almost looks like a "gaussian distribution" with a maximum at the age category "020" (20-39 years old), see table 3.2. This fact makes the training of the neural networks less efficient.

4.3 Results from the three best models for gender detection

Model	Important settings (for the training of the model)	Accuracy (from the last saved model)	Accuracy (from the best epoch)
Model 16	Layers: hdl1(1024), hdl2(4024). lr = 0.0005, l2(0.01), opt = Adam. EarlyStopping(patience = 10). ReduceLROnPlateau(factor = 0.5).	0.9682529	0.9664049
Model 18	Layers: hdl1(4024), hdl2(1024). lr = 0.0005, l2(0.01), opt = Adam. EarlyStopping(patience = 10). ReduceLROnPlateau(factor = 0.5).	0.9677249	0.9638308
Model 22	Layers: hdl1(1024), hdl2(4024), hdl3(1024). lr = 0.0005, l2(0.01), opt = Adam. EarlyStopping(patience = 10). ReduceLROnPlateau(factor = 0.5).	0.9662069	0.9683849

Table 4.3. The three best models for gender detection are shown in the table. The callbacks that are used are early stopping and "ReduceLROnPlateau" with the "scale factor" that the learning rate, lr, is multiplied with (patience is here 3). An l2 regularization is used in all hidden layers with the "strength" given by the value in parenthesis. The notation "hdl1" means that it is the first hidden dense layer with the number of neurons that is given by the value inside the parenthesis.

The three best models for detecting *gender* and their accuracies can be seen in table 4.3. The best model to detect gender has an accuracy score of 0.96838 which is better than our goal in this project of an accuracy score of 0.95. All models in the table have an accuracy that exceeds the goal of 0.95. A lot of the trained models also have an accuracy above 0.95. Most of the models are consistent in the sense that they are stable and gave similar results when they predicted the gender from images.

Slow training with a low learning rate and many epochs gave the best results. For all the trained models to detect gender some level of overfitting is seen, see the graphs for model 22 in fig 4.5 as an illustration of this effect.

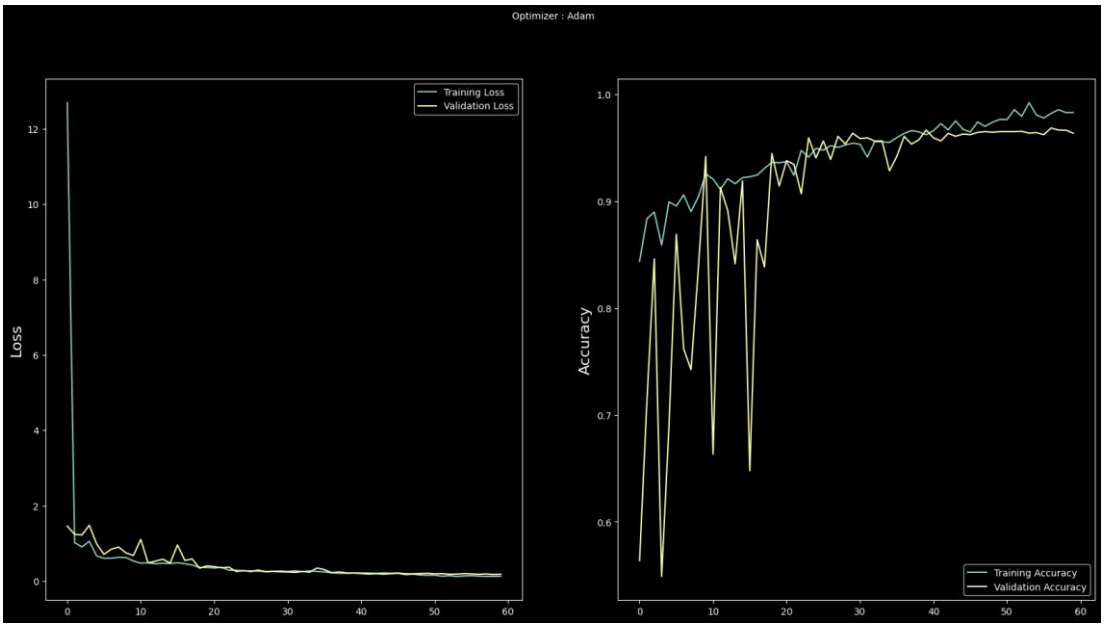

Figure 4.5. The history of the loss and the accuracy during training for the best model (model 22) for gender detection is shown in the graph. The yellow line is the loss or accuracy on the validation data and the blue line is the loss or accuracy on the training data.

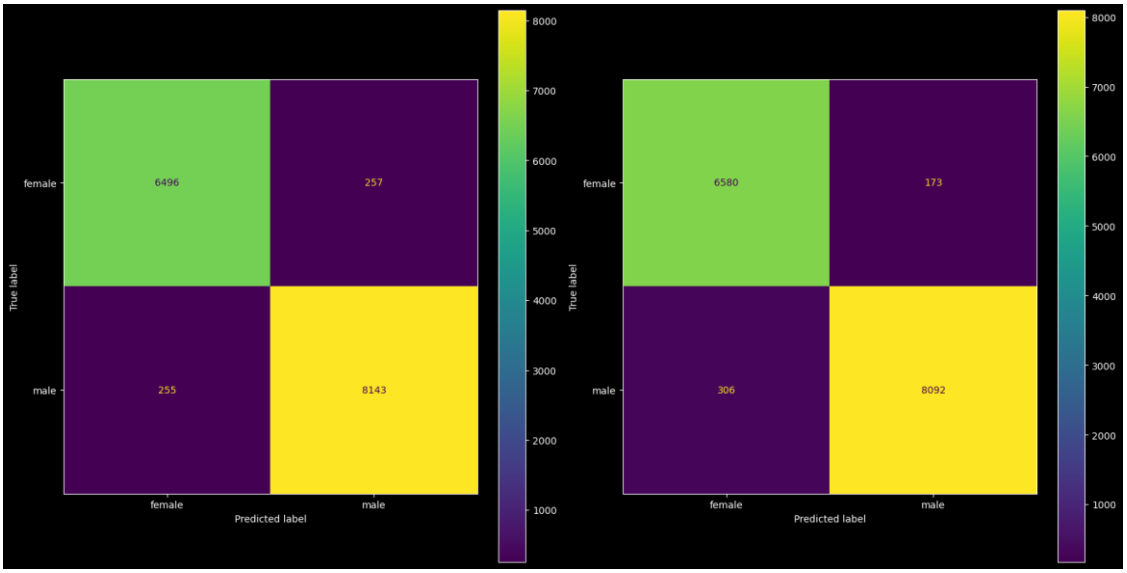

Figure 4.6. The confusion matrices for the best model (model 22) for gender detection are shown in the graph. On the right the confusion matrix for model 22 for the last saved model can be seen and on the left the confusion matrix for model 22 for the best epoch can be seen.

The confusion matrices for gender detection in fig 4.6 illustrate the fact that different simulations give different confusion matrices, which can be seen by looking at the off diagonal elements that are

clearly different in the right and in the left confusion matrix. The number of images in each gender category is somewhat evenly distributed between the gender categories, see table 3.3, and there are only two categories that contain a lot of training data, why the training of these neural networks can be done more efficiently in this case than in the other cases (when detecting emotions or the age category).

The statistical probability to guess the right category or class is given by 1 / (number of classes). This means that the probability of guessing the right emotion is 14 %, the probability of guessing the right age category is 20 % and the probability of guessing the right gender is 50 %. Compare these probabilities with the accuracy scores that are received from the neural networks: 68 % for detecting emotions, 74 % for detecting the age categories and 97 % for detecting the gender, which shows that the neural networks are doing a very impressive job indeed. For a human being it is probably most difficult to guess the right age of an individual, especially an age that lies between two age categories. To guess the gender or the emotion, an individual is displaying, is probably much easier.

In the beginning of this project one age category is used for each age, which lead to 102 classes (where there are gaps, where certain ages didn't exist). The probability of guessing the right age is lower than 1 % in this case, but the neural network still obtained an accuracy score of 19 %. Another attempt is made where the age categories are divided into ten-year intervals (with the last category as 90 years or older), which lead to 10 classes. The probability of guessing the right age is in this case 10 %, but the neural network obtained an accuracy score of 48 %. Why these results aren't used in this project is that it is desirable to predict emotions, age categories or gender with an accuracy score that is larger than 50 %, if it is going to be used in a real application, so that it makes predictions that are more often right than wrong.

5 Conclusions and Future improvements

5.1 Conclusions

The questions that are asked at the beginning of this project will be answered here. The questions are:

1. Can I create programs in python that train neural networks to identify the emotion, the age category and the gender from the images in the dataset with models that can be saved?
2. Can I create a program in python that loads trained models and then uses these models to identify the emotion, the age category and the gender from the images, which are obtained from a laptop's web camera in real time?
3. Can I achieve an accuracy score of at least 0.65 from one of my trained models when detecting "emotions"?
4. Can I achieve an accuracy score of at least 0.70 from one of my trained models when detecting "age categories"?
5. Can I achieve an accuracy score of at least 0.95 from one of my trained models when detecting "gender"?

The answer to the *first two questions* is that I managed to make the four desired python programs with working code. The first three programs trained the neural network models to detect the emotion, the age category and the gender, and then saved each model as a file. I managed to make a program that could train the model on both a dataset created from a generator and from a dataset where the images are loaded directly from the hard drive and stored as variables in a python program ("Jupiter file"). The stored models are then saved and used in another python program ("main.py"). This program then uses the three saved models to predict the emotion, the age category and the gender from images. Faces are scanned from a laptop's web camera in real time and the emotion, the age category and the gender of the scanned face are obtained. I tested that different emotions could be obtained by making different expressions on my face in front of the web camera. I also tried the program with individuals of different ages and genders, which also worked.

The answer to the *third question* is that the best model to detect emotions achieved an accuracy score of 0.6755 which is better than the desired goal of 0.65. A lot of the trained models received an accuracy score above 0.65, so the goal of 65 % accuracy in this project is fulfilled. The answer to the *fourth* and the *fifth question* are that the desired goal of 70 % accuracy for the "age category" detection is obtained and the desired goal of 95 % accuracy for the "gender" detection is obtained in this project. A lot of the trained models for age and gender detection received an accuracy score above their desired goal. The best model for "age category" detection achieved an accuracy score of 0.7381 and the best model for "gender" detection achieved an accuracy score of 0.9684, which in both cases is better than their desired goal in this project, even if it is only by a small margin. It would be desirable to get a better accuracy score, but to do so better quality of the data and more data would probably be needed.

5.2 Future improvements

There is still a lot to be done to improve the accuracy score. The image data is of a low quality and the emotion labeling of the images doesn't seem to be correct in many instances. The images could also be cropped or framed so that the entire face, but not other things are shown in the images. Augmentation of the images like rotating, flipping, adjusting contrast or adjusting brightness could be used. Images with higher resolutions could be used. Images of a larger size could be used and images with colors could be used (that contains three RGB color channels). One important improvement is to use more datasets than the one that is already used with, for instance, Kaggle. The number of images that exist in the different categories could be distributed more evenly (for instance, there are only few images of the emotion disgust). For colored images transfer learning could be used with pre-trained models. The colored images could make it easier to identify certain aspects in the image like lips are red when the mouth is studied.

Similar ideas for improvements as are done for emotion detection could also be used for detecting age categories and gender. For the age categories the number of images that exist in each category could be distributed more evenly and more training data could be used. Color images are used for detecting age and gender, but grayscale images could be used instead, which might improve the performance of the neural networks. Transfer learning could be used when detecting age and gender, and since color images are used, it is probably easier to find pre-trained models.

The neural network models could also identify other aspects in the images such as clothes, hair colour or ethnicity and be used for other purposes. This would contain a lot of similarities and a lot of the work from this project could be used.

The last python program ("main.py") could also be improved by creating a more user-friendly interface with additional functionalities. The python program ("main.py") could instead be a "web page" or "streamlit application" that uses a web camera to scan faces and identify features such as emotions, age, gender, and other attributes like hair color or ethnicity. Furthermore, additional functionalities, such as the ability to save the identified attributes from the scanned face to a file, could enhance the application.

Other senses, beyond *visual detection*, could be used, such as *sound* or *smell*, particularly for detecting *gender* or *emotions*. For instance, dogs can identify an individual's gender, recognize if they are ill, or sense fear by smelling with their nose. In theory, an electronic smell detector might be able to replicate some of these capabilities. Additionally, the sound of an individual's voice from a video could be analyzed and used to detect the speaker's gender or emotions, such as fear. At least theoretically, this should be possible. By combining the detection methods from the three senses mentioned above, the accuracy score for the prediction of gender or emotions could be improved. Achieving this, however, would probably require advancements in electronic devices, that are capable of scent detection, which are not yet available, but may be developed in the future.

References

[1] Géron, A. (2019). Hands-on Machine Learning with Scikit-Learn Keras & TensorFlow. Second Edition. Canada: O'Reilly.

[2] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional. Advances in Neural Information Processing Systems 25 (pp. 1097–1105). Neural Information Processing Systems Foundation, Inc. (NeurIPS). Retrieved from https://proceedings.neurips.cc/paper/2012

[3] Keras. (2024). Retrieved from https://keras.io/

[4] TensorFlow. (2024). Retrieved from https://www.tensorflow.org/

[5] J. Hertz, A. Krogh and R. G. Palmer (1991). Introduction to the theory of neural computation. Redwood City: Addison Wesley.

[6] M. H. Hassoun (1995). Fundamentals of artificial neural networks. First Edition. Cambridge: MIT Press.