INTRODUCTION TO MACHINE LEARNING

FINAL REPORT

FACE AND GENDER RECOGNITION

TEAM MEMBERS:

NAZIRA TUKEYEVA

PANPAN LIU

HUGO LEMASSON

IVAN STANKEVICH

CONTENT

INTRODUCTION

The goal of this project is to implement a machine learning model for face and gender recognition using various approaches and techniques. We chose this topic as it is a widely used technology in many industries such as security, marketing, and human resources. Our aim is to develop a model that can accurately recognize faces and determine gender in real-time. Through extensive testing and analysis, we hope to achieve a high level of accuracy and precision in our results.

In this report, we present a face recognition and gender classification system using computer vision and deep learning techniques. The system uses a combination of two face detection methods, MTCNN and Cascade Classifier, to detect and align faces in images. The detected faces are then passed through a deep learning model, built using Keras, to classify the gender of the person in the image. The system also includes a real-time component that allows for detection and classification of faces in video streams from a webcam. The performance of the system is evaluated using various metrics such as accuracy, precision, recall, and F1 score, and the results are analyzed to understand the effectiveness of the system for different genders. Overall, the report aims to present a comprehensive overview of the design, implementation, and evaluation of a face recognition and gender classification system.

The dataset used for this project consisted of images of girls and boys. 70% of the data was used for training and 30% for testing. The testing data was split equally, with half of the images being of girls and the other half being of boys. The dataset was used to train a deep learning model using the Keras library, which was then used to detect the gender of faces in images and in real-time video streams. The four main metrics used to evaluate the classification model were accuracy, precision, recall, and F1 score. The results showed that the model was effective in detecting the gender of individuals, with an overall accuracy of 94%, precision of 93%, recall of 93%, and F1 score of 93%. The model performed slightly better in detecting girls than boys, with higher accuracy, precision, and recall for girls compared to boys.

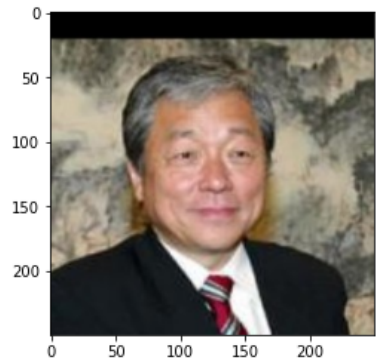
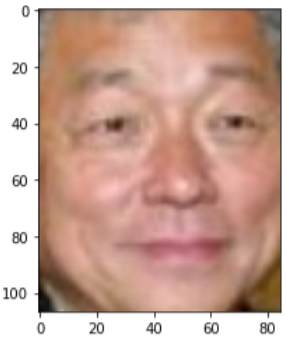
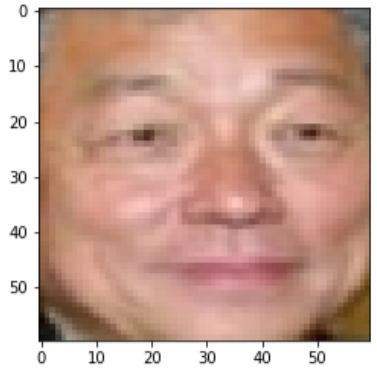
FACE RECOGNITION

The first primary task for this project is to implement the recognition of the face on image. There are multiple different approaches to detect faces. Hence, in this part I will describe the methods I used.

Approach 1: Face detection using MTCNN.

This framework is designed to implement both face recognition and face alignment tasks. The idea behind using this framework in our project was the fact that it can detect partially covered faces as well as recognize multiple faces in our image. The detector returns the image that formatted as the bounding box. The bounding box is formatted as [x, y, width, height] under the key 'box'.

The example of using MTCNN detector is shown below:

		
Original image (250,250,3)	Bounding box (107,85,3)	Reshaped image (60,60,3)

However, in order to process more than 27 thousand of images from our dataset this approach is considered to be time consuming.

Approach 2: Cascade Classifier from cv2 module.

Cascade Classifier is a machine learning based approach where a cascade function is trained from a lot of positive and negative images. Further it's used to detect objects in other images.

For this project, we used a pre-trained Haar-Cascade Classifier. It is embodied in the cv2.CascadeClassifier class. In order to return the bounding box in the form of a rectangle we used detectMultiScale() method which detects an object on an image and returns its coordinates as tuples.

As a result, we print out the number of faces in an image and the face with the bounding box around the face in a separate window.

Several examples of using this model:

1. Example 1: 1 face detection

jupyter Face Recognition Model (unsaved changes)

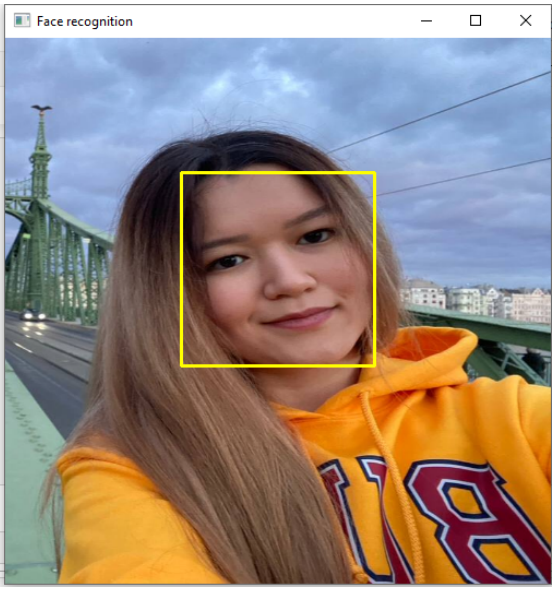
File Edit View Insert Cell Kernel Widgets Help

Run Code

```
In [*]: import cv2

img = cv2.imread('nazira.jpeg')
img=cv2.resize(img,(500,500))
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

face_cascade = cv2.CascadeClassifier('face_detection.xml')

faces = face_cascade.detectMultiScale(gray, 1.1, 2)
print('Number of detected faces:', len(faces))\

for (x,y,w,h) in faces:

   cv2.rectangle(img,(x,y),(x+w,y+h),(0,255,255),2)

cv2.imshow('Face recognition',img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

```
Number of detected faces: 1
```

```
In [ ]:
```



2. Example 2: several faces detection

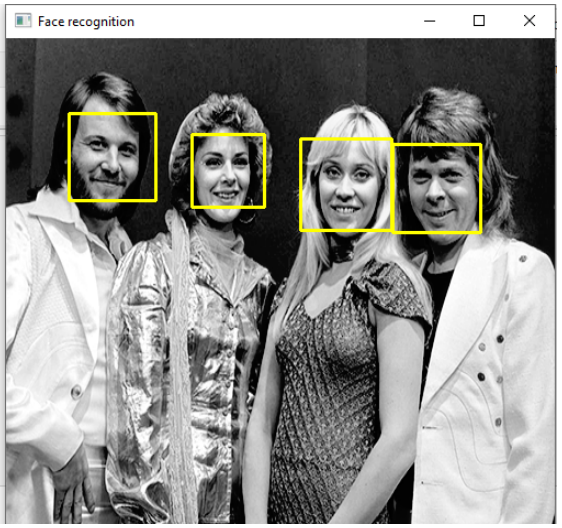
jupyter Face Recognition Model (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help

Run Code

```
In [*]: import cv2

img = cv2.imread('img1.png')
img=cv2.resize(img,(500,500))
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

face_cascade = cv2.CascadeClassifier('face_detection.xml')

faces = face_cascade.detectMultiScale(gray, 1.1, 2)
print('Number of detected faces:', len(faces))\

for (x,y,w,h) in faces:

   cv2.rectangle(img,(x,y),(x+w,y+h),(0,255,255),2)

cv2.imshow('Face recognition',img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

```
Number of detected faces: 4
```



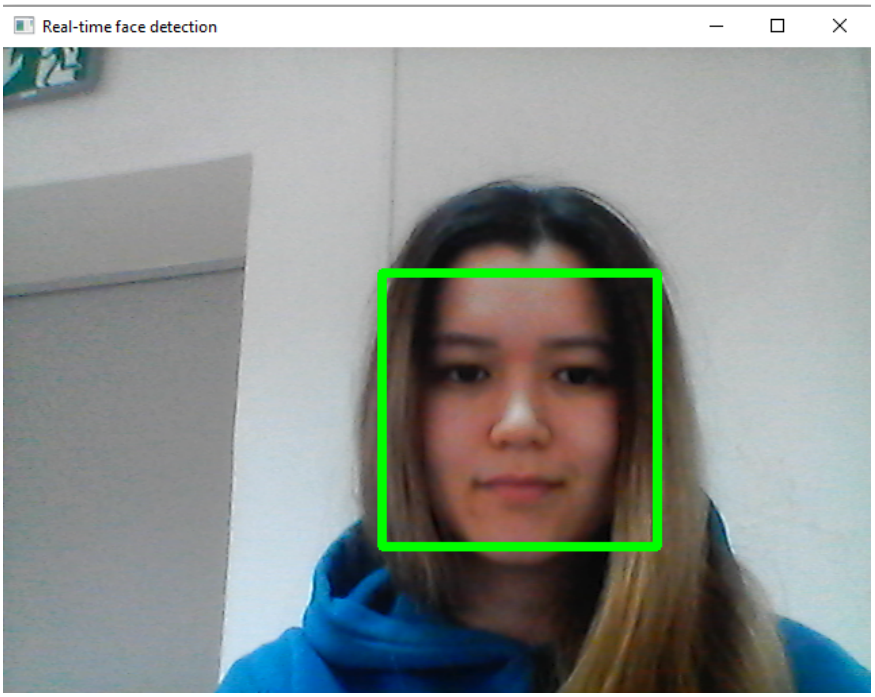
Further, we decided to use this model as the results are quite accurate as can be seen in the examples above.

REAL TIME FACE RECOGNITION

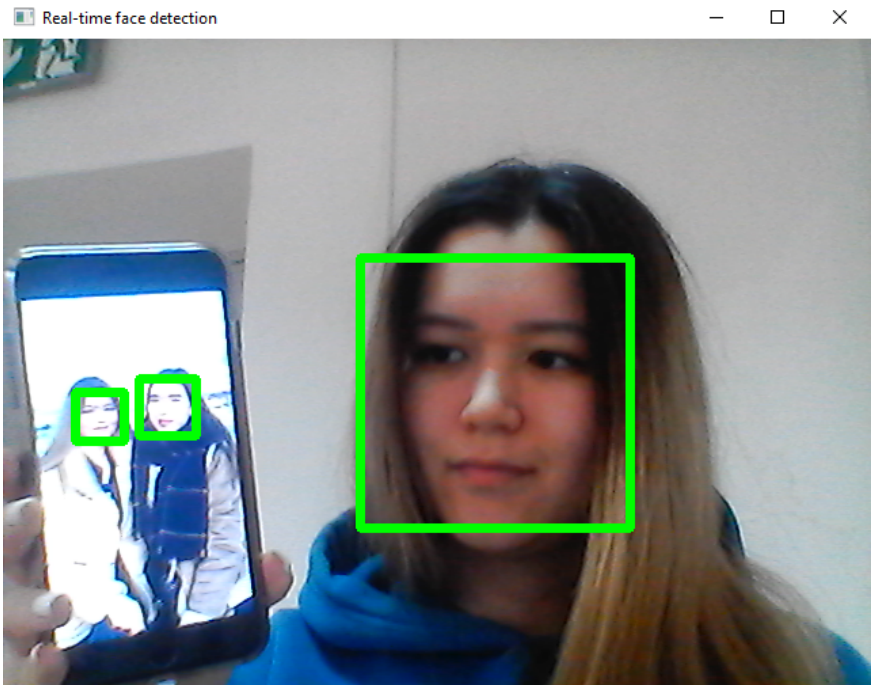
Next on, I adapted the program so that it works as a real-time face detection. It is the same approach but instead of inputting the image, we get a video stream from a webcam. To take the input from the webcam, I made a change to the VideoCapture() call.

The example of the result of real-time face recognition is below:

1. Example 1: single face



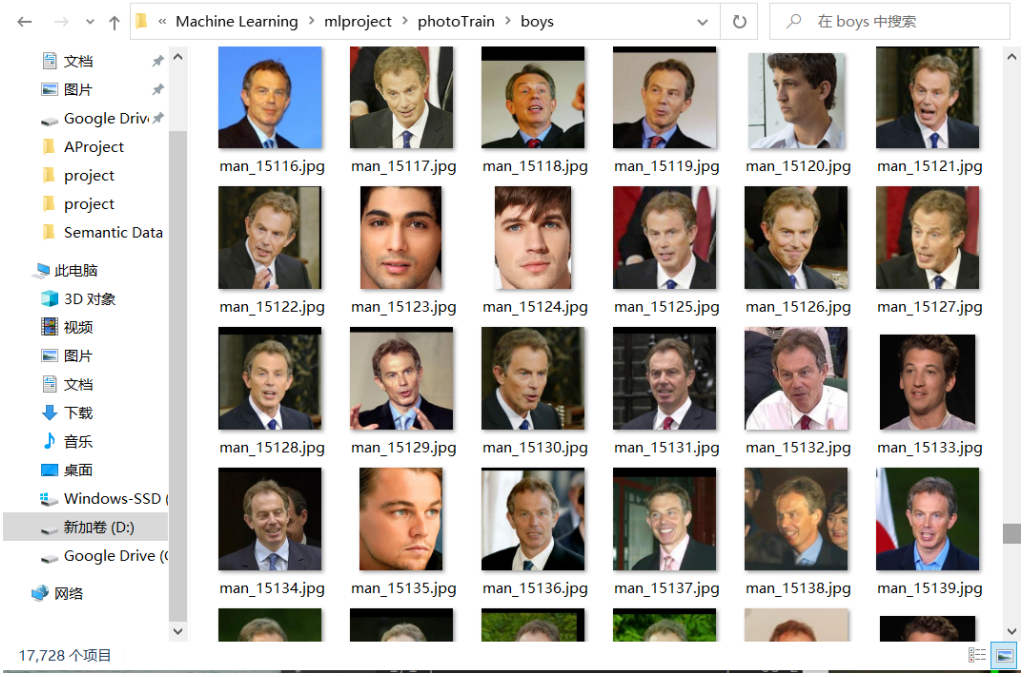
2. Example 2: multiple faces

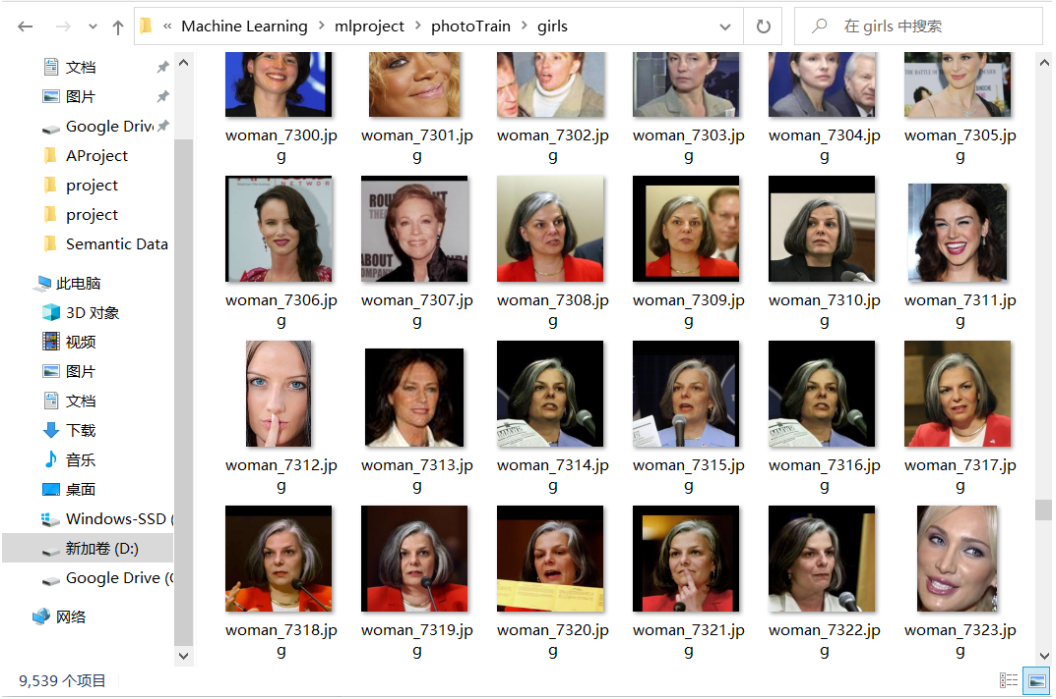


GENDER RECOGNITION

After the process of face recognition, what we should do next is to determine gender for each face we detected. The approach I use is model Sequential in the python deep learning library Keras.

Datasets for training: 17728 men and 9539 women face photos.





Steps for model training:

1. Prepare training faces and their corresponding gender labels.
   Read the raw photos and detect the faces in every photo by cv.CascadeClassifier with our face detection model 'face_detection.xml'. After that, grey face pictures in the bounding box are saved in an array list and labels, 0 for women and 1 for men are created in the list as well.
2. Build sequential mode. There are three layers in our model: Flatten layer, Dense layer with 'relu' activation, Dense layer with 'softmax' activation.

```
model = tf.keras.models.Sequential([
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(10, activation='softmax')
])
```

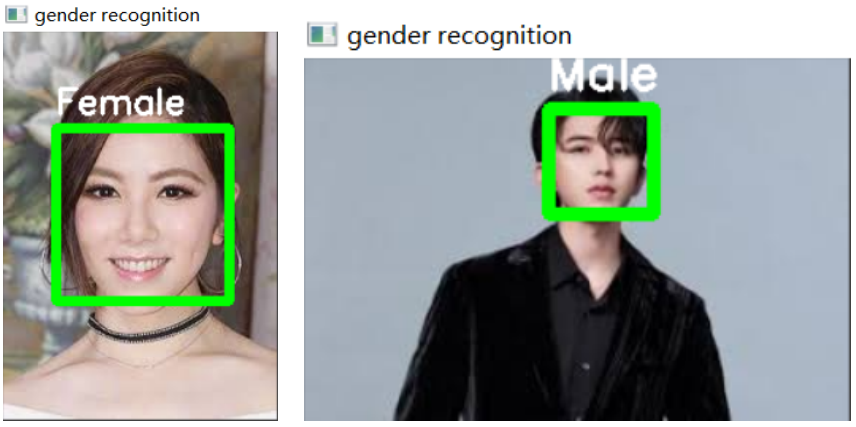
3. Configure sequential mode with following parameters.

```
model.compile(
    optimizer='adam',
    loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=False),
    metrics=['sparse_categorical_accuracy']
)
```

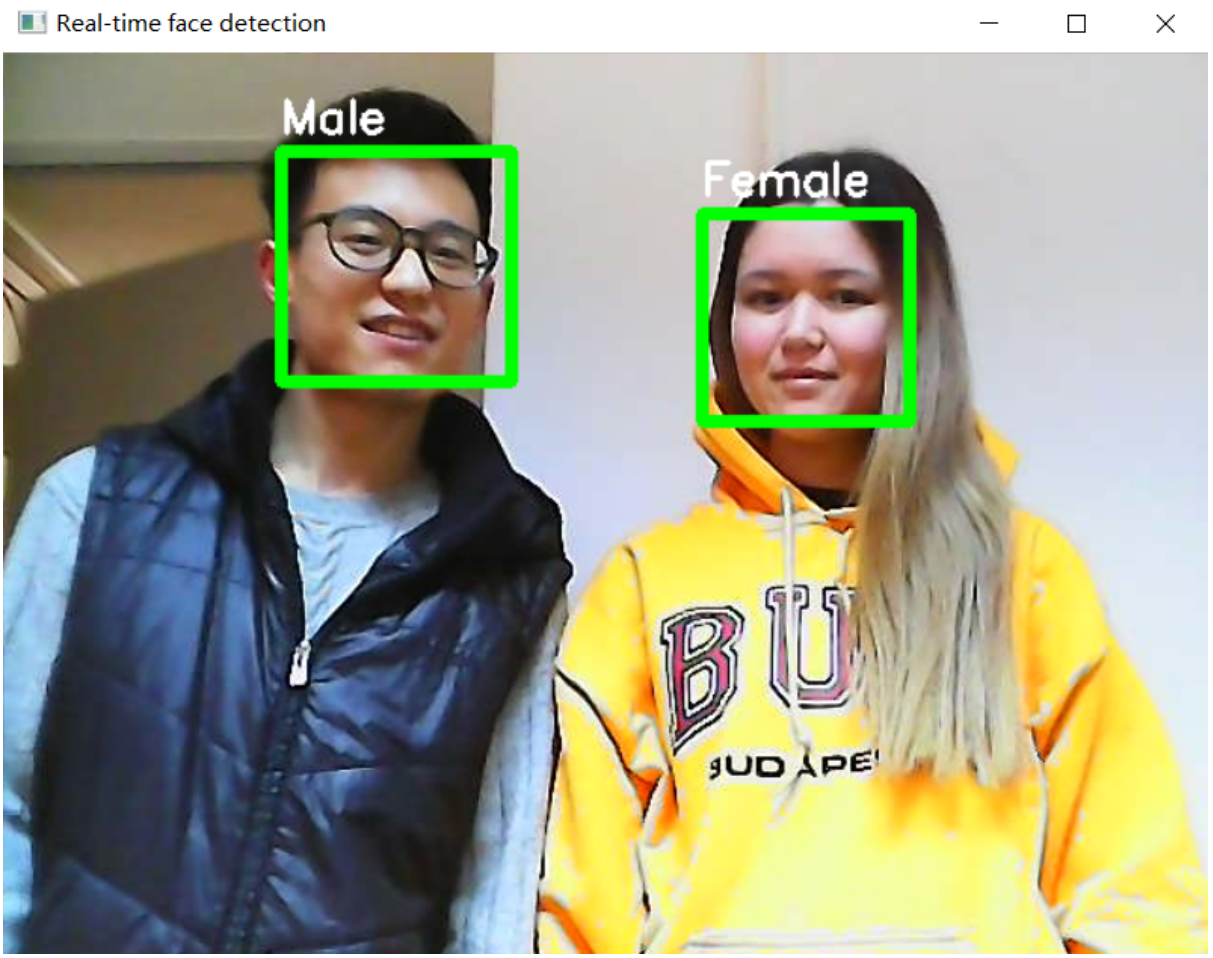
4. Feed the data in the model.

```
model.fit(
    facesArray,
    idsArray,
    batch_size=32,
    epochs=500,
    validation_data=(x_test,y_test),
    validation_freq=1
)
```

Face recognition for pictures:



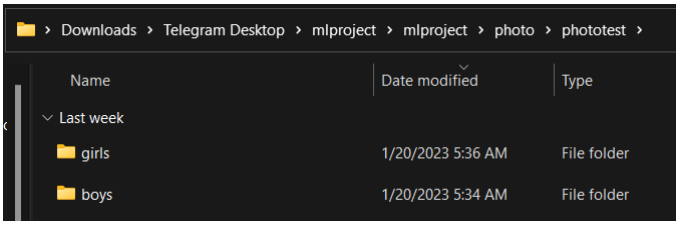
Real-time face recognition:


Real-time face detection
Male
Female

EVALUATION

For this project we used a dataset with images of girls and boys, we used 70% of the data for training and 30% of the data for testing. Testing data is split equally, the test data is split equally, half boys and half girls.

To evaluate the model the images of boys and girls are in two folders, respectively “boys” and “girls”

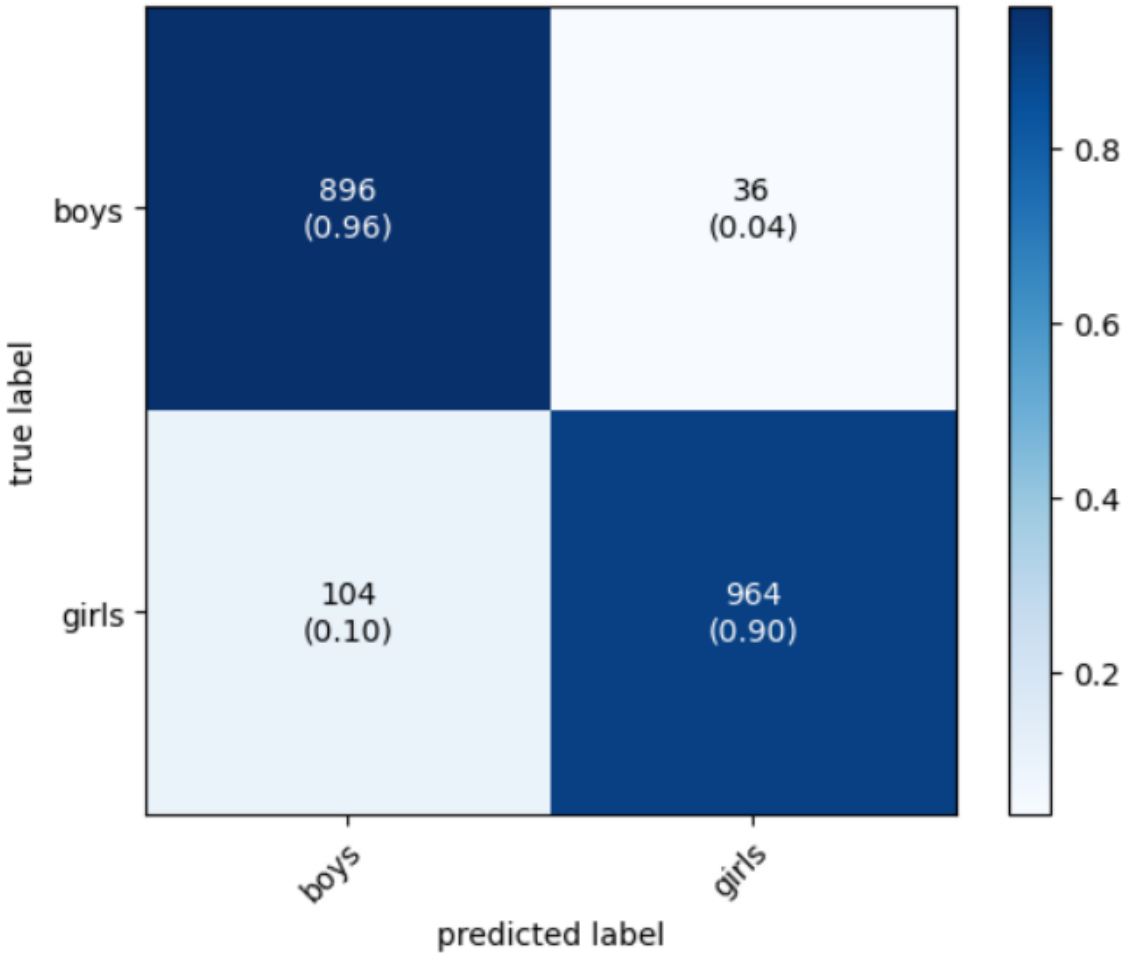



```
path = 'C:/Users/hugol/Downloads/Telegram Desktop/mlproject/mlproject/photo/phototest'
categories = os.listdir(path)
for category in categories:
    folder_path = os.path.join(path, category)
    imageNames = os.listdir(folder_path)
    for imageName in imageNames:
        imagePath = os.path.join(folder_path, imageName)
        img = cv.imread(imagePath)
        gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
        faces = face_classifier.detectMultiScale(gray)
        for (x,y,w,h) in faces:
            face_img = gray[y:y+w, x:x+w]
            resized = cv.resize(face_img, (60,60)) / 255.0
            resized = np.reshape(resized, (1,60,60,1))
        result = model.predict(resized)
        label = np.argmax(result, axis=1)[0]
        if(category=='girls'):
            if(label==0):
                girls_TP+=1
                boys_TN+=1
            else:
                girls_FN+=1
                boys_FP+=1
            Nb_girls+=1
        elif(category=='boys'):
            if(label==1):
                boys_TP+=1
                girls_TN+=1
            else:
                boys_FN+=1
                girls_FP+=1
            Nb_boys+=1
```

We count differently the true positive, true negative... of boys and girls to be able to compare them.

First, we plot the confusion matrix:

```
binary = np.array([[boys_TP, boys_FP],
                   [boys_FN, boys_TN]])

fig, ax = plot_confusion_matrix(conf_mat=binary,
                                show_absolute=True,
                                show_normed=True,
                                colorbar=True,
                               class_names=['boys','girls'])
plt.show()
```



As we can see, our model is pretty good at detecting if a person is a girl or a boy, only 7% of the data is not predicted correctly.

Analysis of the model quality

The four main metrics used to evaluate our classification model are accuracy, precision,recall and F1 score.

For each metric we will analyze the result in 3 different ways, first on the girls half then with the men then with everything (boys+girls) , so we will be able to compare the effectiveness of our model on girls and boys.

```
#print the result
print("boys TP: " + str(boys_TP))
print("boys FN: " + str(boys_FN))
print("boys FP: " + str(boys_FP))
print("boys TN: " + str(boys_TN))

#accuracy=number of correct prediction/total Number of prediction
girls_accuracy=girls_TP/Nb_girls
boys_accuracy=boys_TP/Nb_boys
print("girls accuracy: "+ str(girls_accuracy))
print("boys accuracy: "+ str(boys_accuracy ))

accuracy= (girls_TP + boys_TP)/(Nb_girls+Nb_boys)
print("Overall accuracy: "+ str(accuracy))

#precision=TP/TP+FP
girls_precision = girls_TP/(girls_TP+girls_FP)
boys_precision = boys_TP/(boys_TP+boys_FP)
print("girls precision: "+ str(girls_precision) )
print("boys precision: "+ str(boys_precision) )

precision= (boys_TP + girls_TP)/(boys_TP + girls_TP +girls_FP+boys_FP)
print("Overall precision: "+ str(precision))

#recall=TP/TP+FN
girls_recall = girls_TP/(girls_TP+girls_FN)
boys_recall = boys_TP/(boys_TP+boys_FN)
print("girls recall: "+ str(girls_recall) )
print("boys recall: "+ str(boys_recall) )

recall = (boys_TP + girls_TP)/(boys_TP + girls_TP +girls_FN+boys_FN)
print("Overall recall: "+ str(recall))

#F1_score=2*(precision*recall)/(precision+recall)
girls_F1_score = 2*(girls_precision*girls_recall)/(girls_precision+girls_recall)
boys_F1_score = 2*(boys_precision*boys_recall)/(boys_precision+boys_recall)
print("girls F1_score: "+ str(girls_F1_score) )
print("boys F1_score: "+ str(boys_F1_score) )

F1_score = 2*(precision*recall)/(precision+recall)
print("F1 score: "+ str(F1_score))
```

Accuracy:

Accuracy is defined as the percentage of correct predictions for the test data. It can be calculated easily by dividing the number of correct predictions by the number of total predictions.

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}$$

Girls accuracy = 0.964

Boys accuracy = 0.894

Overall accuracy = 0.93

The girls accuracy is a little better than the boys accuracy, it means our model detects girls more easily than boys.

Precision and recall are useful in cases where classes aren't evenly distributed, which is not our case because we have as many girls as boys in our dataset, moreover the true positives and the false negatives have the same importance for gender detection, they are both relevant.
We will still show you their value because we can understand the biases of our model:

Precision:

Precision is defined as the fraction of relevant examples (true positives) among all of the examples which were predicted to belong in a certain class.

$$\text{Precision} = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

Girls precision = 0.903

Boys precision = 0.961

Overall precision = 0.93

The boys precision is better than the girls precision, this means that our model is more likely to predict that the person is a girl rather than a boy.

Recall:

Recall is defined as the fraction of examples which were predicted to belong to a class with respect to all of the examples that truly belong in the class.

$$\text{Recall} = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

Girls recall = 0.964

Boys recall = 0.869

Overall recall = 0.93

The girls recall is better than the boys recall, because if our model is more likely to predict that the person is a girl rather than a boy, then it’s normal to better predict a person to belong to a girl class with respect to all of the examples that truly belong in the class.

F1 score:

F1 score is an average of Precision and Recall, it means that the F1 score gives equal weight to Precision and Recall

$$F1\ score = 2 * \frac{Precision\ * Recall}{Precision + Recall}$$

Girls F1 score = 0.932

Boys F1 score = 0.928

Overall F1 score = 0.93

F1 score is high because both Precision and Recall are high.

WORK BREAKDOWN

TASK	TEAM MEMBER
Introduction & Dataset description	Ivan
Face Recognition - Methods 1 & 2	Nazira
Real-Time Face Detection	Nazira
Gender Recognition	Panpan
Real-Time Gender Detection	Panpan
Evaluation	Hugo
Conclusion & Discussions	Ivan
User Manual	Ivan & Nazira
Final Report	Each member reports the description of their part (above mentioned tasks)

DISCUSSION ON RESULTS

In this report, you have described the process of implementing a face recognition and gender classification system using multiple approaches. The first approach you used was the Multi-task Cascaded Convolutional Networks (MTCNN) for face detection. This method is known for its ability to detect partially covered faces and recognize multiple faces in an image. However, you mentioned that this approach may be time-consuming when processing a large dataset. As an alternative, you also implemented the use of a pre-trained Haar-Cascade Classifier from the cv2 module for face detection. This machine learning-based approach uses a cascade function trained on a dataset of positive and negative images to detect objects in other images. The detectMultiScale() method was used to return the coordinates of the bounding box around the face in the form of a rectangle. You also adapted the program to work as a real-time face recognition system by using a video stream from a webcam as input. For gender classification, you used a Sequential model from the Keras deep learning library. You prepared training faces and their corresponding gender labels, and used the Cascade Classifier to detect faces in the images. The grey face pictures in the bounding box were then saved in an array list and labels were created, with 0 for women and 1 for men. The model consisted of three layers: a Flatten layer, a Dense layer with 'relu' activation, and a Dense layer with 'softmax' activation. For evaluation, you used a dataset of images of girls and boys, with 70% of the data used for training and 30% of the data used for testing. The testing data was split equally, half boys and half girls. The confusion matrix showed that the model was pretty good at detecting if a person is a girl or a boy, with only 7% of the data not predicted correctly. You also analyzed the model's quality using four main metrics: accuracy, precision, recall, and F1 score. Overall accuracy of the model was 0.93, precision was 0.93, recall was 0.93, and F1 score was 0.93. The girls accuracy was a little better than the boys accuracy, boys precision was better than the girls precision, girls recall was better than the boys recall, and boys F1 score was better than the girls F1 score. In conclusion, the model implementation you described in this report appears to be a promising approach for face recognition and gender classification. The use of the Haar-Cascade Classifier for face detection and the Sequential model from Keras for gender classification resulted in high accuracy and precision. However, it is important to note that the performance of the model may vary depending on the dataset used and further testing and optimization could improve the performance.

CONCLUSION

In conclusion, the project was successful in implementing face and gender recognition using machine learning techniques. The face recognition was achieved using two approaches, MTCNN and Cascade Classifier, with the latter being chosen as it was found to be more accurate. The program was also adapted to work in real-time with a webcam. For gender recognition, a deep learning model was trained using Keras and the model was found to have an overall accuracy of 93%. The model was found to have slightly better performance in recognizing females and had slightly more bias towards predicting females. Overall, the project demonstrates the effectiveness of machine learning techniques in face and gender recognition and has potential applications in various fields such as security, surveillance, and marketing.

USER MANUAL

The instruction on how to launch, configure, and use the solution is below:

1. Download the following files: main.py, evaluation.py, face_detection.xml and gender_recognition.h5. Make sure you are storing them in the same directory.

2. Install a compiler for Python (one of the following):
- PyCharm: https://www.jetbrains.com/pycharm/
- Anaconda: https://www.anaconda.com/products/distribution/

3. Installing the required packages in Python:
- For PyCharm: you need to go to PyCharm, temporarily create a project and open Terminal at the bottom
- For Anaconda: open Anaconda Prompt
In an open terminal or console, enter the following commands:
pip install cv2
pip install keras
pip install pandas

4. Open main.py:
- For PyCharm: File->Open, find and select main.py
- For Anaconda: Open Jupyter Notebook and File->Open, find and select main.py

5. 2 case scenarios:
For real-time detection: Run the code and wait. The camera will appear and you will see the result.
For detection by image: uncomment the lines [71, 83]. On line 71, provide a path to the image as in example. Please, comment the code below line 85, i.e. [85, 112]. Run the code and wait. Once the execution is completed, you will get the result.