
Arbeitsheft:

Wie “erkennt” ein Computer Gesichter?

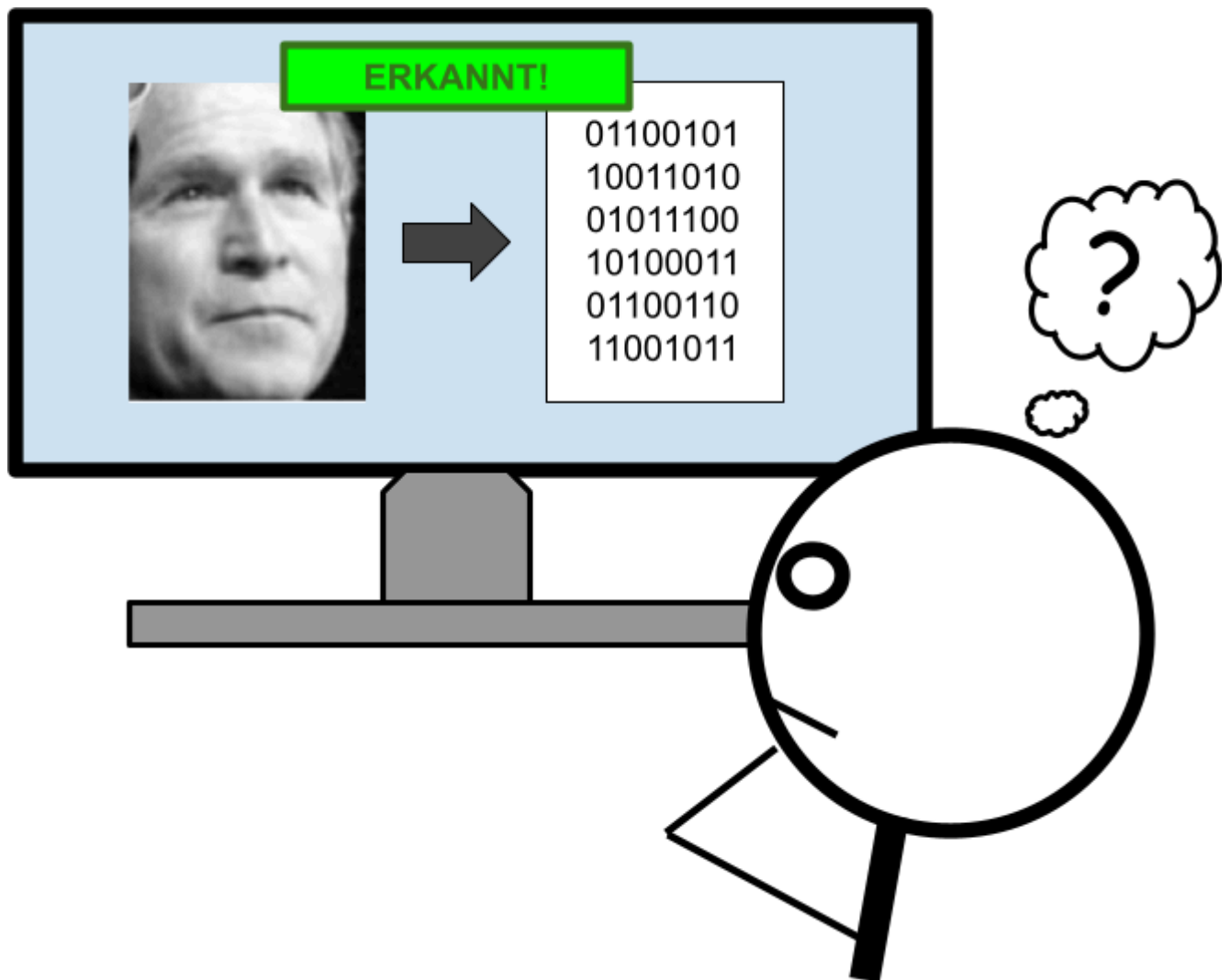


Abb. 1: Gesicht von George W. Bush (LFW Dataset (2007))

Name: _____

Datum: _____

Aufgaben, welche das Niederschreiben von Text in dieser PDF-Datei benötigen, können mithilfe von Kommentaren oder Textfeldern des PDF-Programms beantwortet werden. Zusätzlich kann mit verschiedenen Farben in der PDF-Datei gezeichnet werden.

1. Einleitung

Hast du dich jemals schon gefragt, wie ein Computer in der Lage ist, Gesichter zu erkennen?

Ein Computer macht nur das, was ihm ein Mensch, in Form eines Programmes, befohlen hat. Aber wie fängt man an, einem Computer überhaupt den Befehl zu geben, ein Gesicht zu erkennen?

Sagt man: "Bestimme die Breite der Nase.", dann stellen sich die Fragen: "Was ist eine Nase?" und "Wo auf dem Bild ist die Nase?". Diese Fragen sind für uns Menschen sehr einfach zu beantworten, hingegen ein Programm zu schreiben, was diese "einfachen" Fragen beantwortet, stellt sich als große Herausforderung dar.

Ziel dieses Arbeitsheftes ist es, die grundlegende Funktionsweise der Gesichtserkennung zu vermitteln und dabei auf den gesellschaftlichen Einfluss einzugehen, welche derartige Algorithmen haben können. Zuerst wird darauf eingegangen, wo uns Gesichtserkennung bereits im Alltag begegnet. Dann ist es Ziel, die technischen Grundlagen kennenzulernen und diese dann in vereinfachter Form zu implementieren. Anschließend werden diese Grundlagen an echten Bildern getestet und am Schluss wird die gesellschaftliche Wechselwirkung nochmal genauer betrachtet.

Gesichtserkennung taucht in der heutigen Welt vermehrt auf und wird verwendet, um sich zu identifizieren oder um identifiziert zu werden. Dabei hat man selber nur teilweise einen Einfluss darauf, ob man richtig oder falsch identifiziert wird und ob Entscheidung für oder gegen einen selbst getroffen werden. Dementsprechend ist es wichtig, die grundlegende Funktion dieser Verfahren zu kennen und sich über die Schwächen dieser bewusst zu sein. Vieles wird heutzutage über künstliche Intelligenz geregelt, dessen Begründung bezüglich einer Identifizierung schwierig nachzuvollziehen ist. Hingegen können viele der zum Machine Learning gehörenden Verfahren auch ohne große Tiefe betrachtet werden.

2. Gesellschaftlicher Einfluss

Einige Beispiele für Gesichtserkennung sind:

- Eine Sicherheitskamera am Flughafen erkennt verdächtige Personen.
- In einem Geschäft wirst du von einem Werbeschild „wiedererkannt“ und passende Werbung erscheint.

2.1 Sammel stichpunktartig weitere Beispiele aus deinem Alltag, in denen du mit Gesichtserkennung konfrontiert wirst oder du diese verwenden (z.B. Smartphone).

2.2 Nenne zu mindestens drei der folgenden Problemfelder, jeweils ein kurzes Beispiel oder einen Grund, warum diese problematisch sein könnten in Bezug auf automatische Gesichtserkennung.

Diskriminierung	
Fehlklassifikation	
Überwachung	
Datenschutz	
Manipulation	

Der Einsatz von Gesichtserkennung an öffentlichen Einrichtungen wird kontrovers diskutiert.

2.3 Beantworte die Fragen zu den folgenden Fallbeispielen stichpunktartig.

Fallbeispiel 1 (Gesichtserkennung in der Schule):

An einer Schule wird testweise Gesichtserkennung zur automatischen Anwesenheitskontrolle eingesetzt. Morgens wird beim Betreten des Schulgebäudes das Gesicht gescannt und die Anwesenheit automatisch registriert.

- Welche Vorteile/Nachteile könnte dies haben?

Fallbeispiel 2 (Gesichtserkennung an Umschlagplätzen):

In einer Großstadt werden Bahnhöfe und andere Umschlagplätze mit Gesichtserkennungskameras überwacht, um bestimmte Personen schneller zu finden.

- Wie könnte das die Sicherheit erhöhen?
- Welche Bedenken könnten dabei entstehen?

3. Wie Maschinen lernen, Gesichter zu erkennen

Ein Computer "sieht" keine Gesichter, sondern verarbeitet algorithmisch Zahlenwerte. Deswegen werden häufig Bilder, welche große Mengen an Pixeln mit verschiedenen Farbwerten enthalten, reduziert auf eine Datenformat, welche besser vergleichbare Daten enthält.

Beispiele für eine derartige Reduktion sind wie folgt dargestellt:



Abb. 2: Von Gesicht zu Vektorekter (Tony Blair, LFW Dataset (2007))

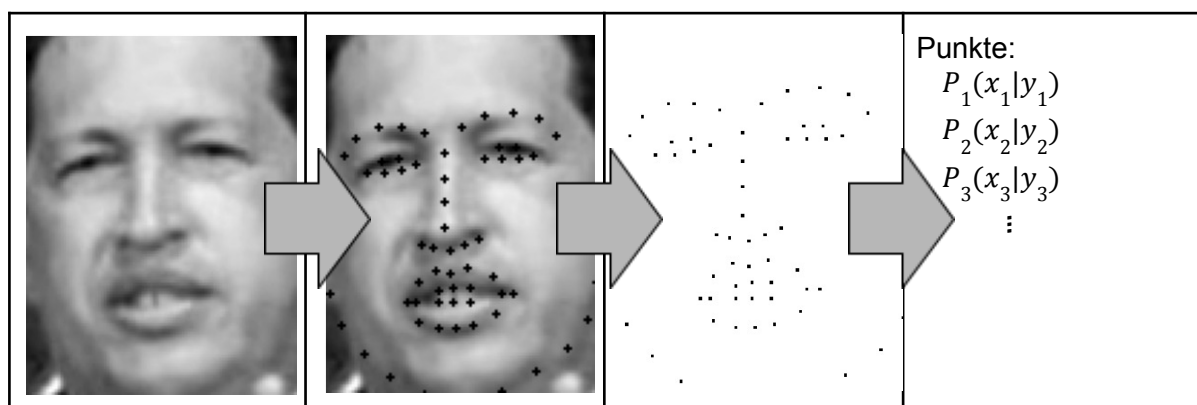
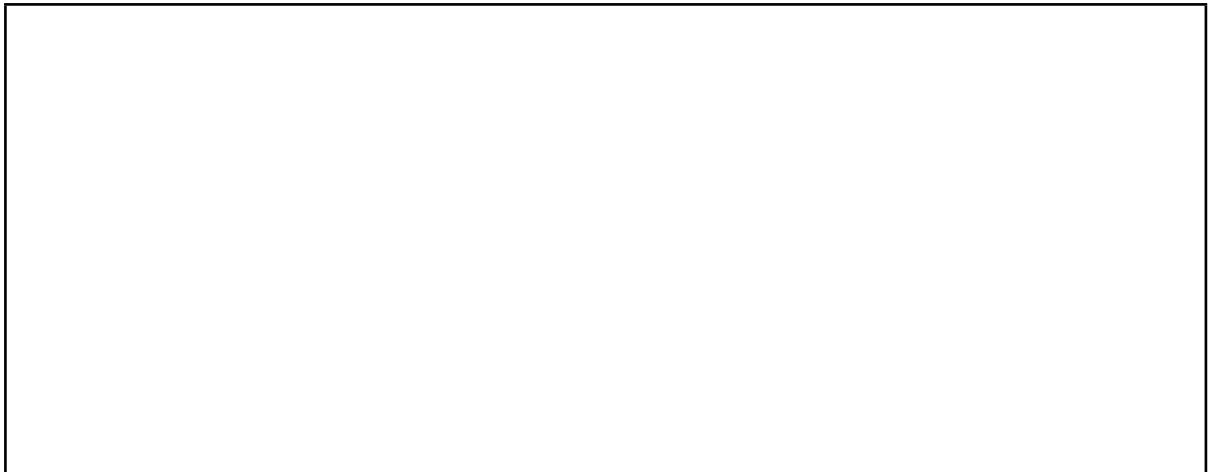


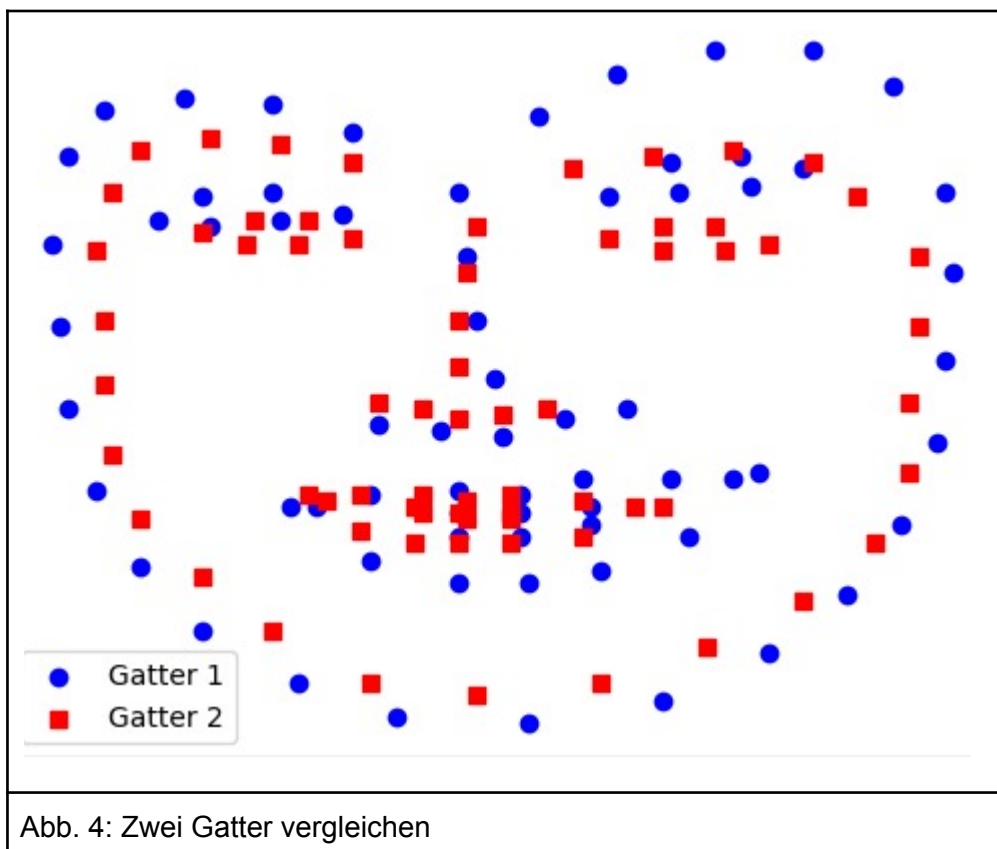
Abb. 3: Von Gesicht zu Punktegatter (Hugo Chávez, LFW Dataset (2007))

3.1 Beschreibe kurz, was du in den obigen Abbildungen 2 und 3 beobachten kannst.

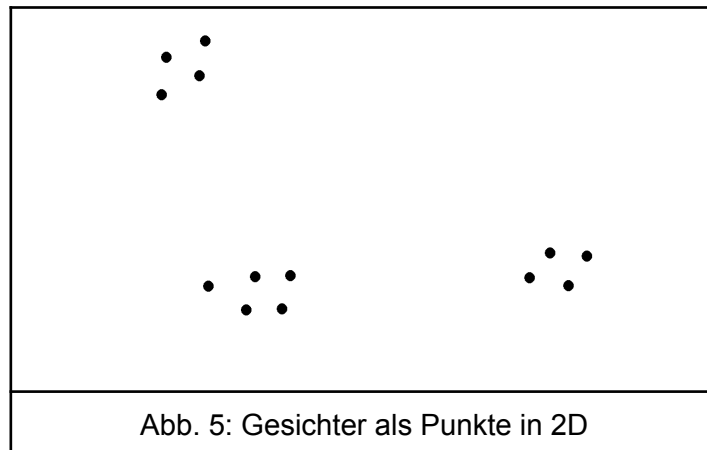


Nun haben wir Werte, die miteinander verglichen werden können. Beispielsweise sollten eine Person A und Person B unterschiedliche Personen sein, wenn sich Werte wie Augenabstand und Nasengröße stark unterscheiden. Ebenfalls sollten zueinander gehörende Punktegatter ähnlich sein.

Jetzt wollen wir weiter darauf eingehen, wie man mithilfe von solchen Werten Gesichter einander zuordnen kann. Dafür müssen diese Werte verglichen werden.

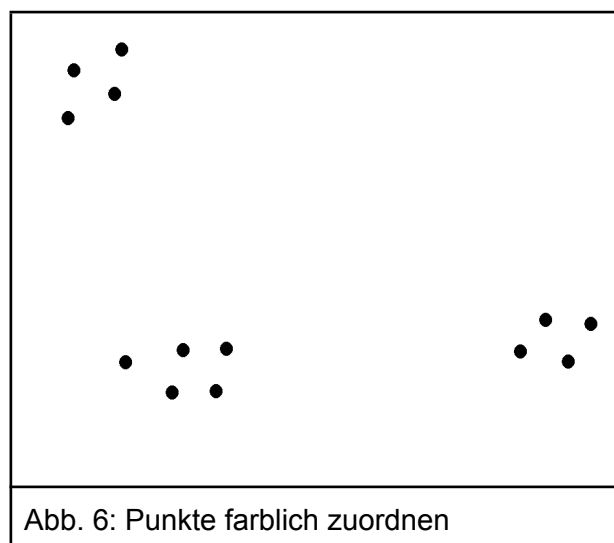


Schon beim Vergleichen von nur zwei Gesichtern ist das Bild überfüllt mit Punkten. Da uns aber nur interessiert, ob ein Bild A ähnlicher ist zu einem Bild B oder einem Bild C, reicht es hier, dass wir anstatt von Punktegattern nur einzelne Punkte betrachten. Der einzige Unterschied zur Realität, der dabei entsteht, ist, dass wir uns nur den Abstand zwischen einzelnen Punkten angucken und nicht den Abstand zwischen den Wertevektoren. Dies macht das Rechnen per Hand und das Nachvollziehen einfacher. Auf technischer Ebene funktioniert der Ansatz in beiden Fällen genauso gut.



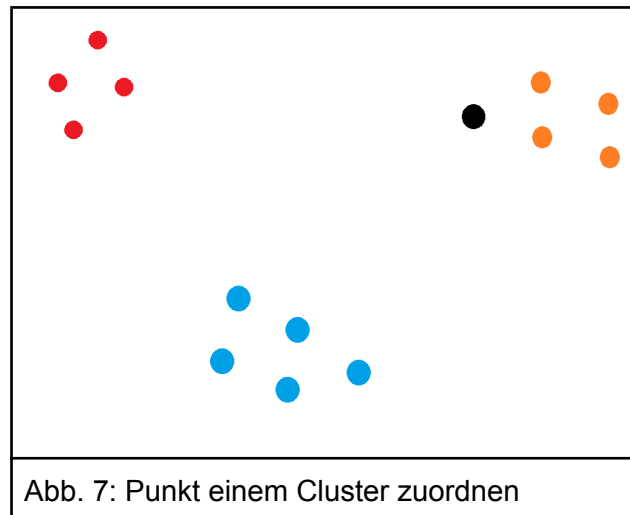
Jeder dieser Punkte in Abbildung 5 repräsentiert ein Bild von einem Gesicht. Ähnliche Gesichter sollten dabei nah nebeneinander sein. Unterschiedliche Gesichter hingegen sind weiter voneinander entfernt.

3.2 Markiere farblich in Abb. 6 die Bereiche, die zu verschiedenen Personen gehören. Zusätzlich **beantworte** die Frage: “Zu wie vielen verschiedenen Personen gehören die Gesichter (Punkte)?”



Antwort: _____

Nun haben wir einen neuen Punkt (schwarz), der einem dieser Cluster (deutsch Haufen/Bündel) zugeordnet werden soll.



3.3 Benenne die Farbe des Clusters, zu dem der schwarze Punkt gehören sollte (Abb. 7), **begründe** zusätzlich kurz, weshalb du diese Farbe gewählt hast.

Wir haben jetzt die drei wichtigsten Schritte der Gesichtserkennung reduziert betrachtet:


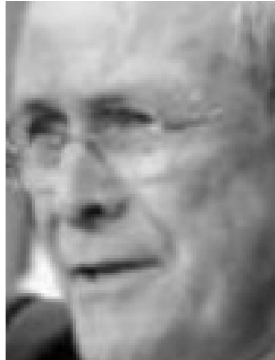


- Das Erstellen eines Wertevektors aus einem Bild (Abb. 2 und 3)
- Das Erstellen von Clustern (Abb. 6)
- Das Zuordnen eines Vektors in ein Cluster (Abb. 7)

Vieles, was wir hier gemacht haben, fällt uns Menschen intuitiv nicht schwer. Hingegen macht ein Computer nur das, was ihm klar beauftragt wurde. Dadurch können in jedem dieser drei Schritte Fehler entstehen, verursacht durch die verschiedensten Gründe. Um diese nachvollziehen zu können, müssen wir uns diese drei Schritte genauer angucken.

4. Vom Bild zum Wertvektor

Um mithilfe eines Computers Bilder auf Werte zu reduzieren, werden häufig künstliche Intelligenzen trainiert, mit einer großen Menge an Trainingsdaten (hier Bilder). Im kommerziellen Bereich werden Datensätze mit weit über eine Million Bildern verwendet, zum Beispiel nutzt Incode Technologies (ein Anbieter für derartige Software) nach eigenen Angaben über 40 Millionen Bilder.

Die folgenden vier Bilder sind deutlich weniger, als normalerweise in der Realität verwendet werden, trotzdem lassen sich daran bestimmte Eigenschaften beobachten.

			
Colin Powell	Donald Rumsfeld	George W. Bush	Gerhard Schröder

Bilder aus dem LFW Dataset (2007)

4.1 Erkenne einige Gemeinsamkeiten, die diese vier Bilder miteinander haben.

4.2 Stelle stichpunktartig eine Vermutung **auf**, warum es nachteilhaft ist, eine künstliche Intelligenz auf ähnlichen Bildern zu trainieren.

Daraus folgern wir bereits die erste Bedingung an ein derartiges Modell (die künstliche Intelligenz). Das Modell muss an einem sehr großen und vor allem diversen Datensatz trainiert werden.

Es gibt aber auch Faktoren, mit denen ein ausreichend trainiertes Modell Probleme hat.

4.3 Hier sind einige Bilder, an denen unser bisher verwendetes Modell gescheitert ist. **Vermute, woran** es gelegen haben könnte.



Abb. 8: Gesicht 1

<https://www.sueddeutsche.de/2022/06/14/3057ee11-806d-4881-87be-e891debb804d.jpeg?q=60&fm=avif&width=1536&rect=0%2C337%2C1024%2C577>



Abb. 9: Gesicht 2

https://www.japantimes.co.jp/uploads/imported_images/uploads/2019/10/f-hkai-a-20191024.jpg



Abb. 10: Gesicht 3

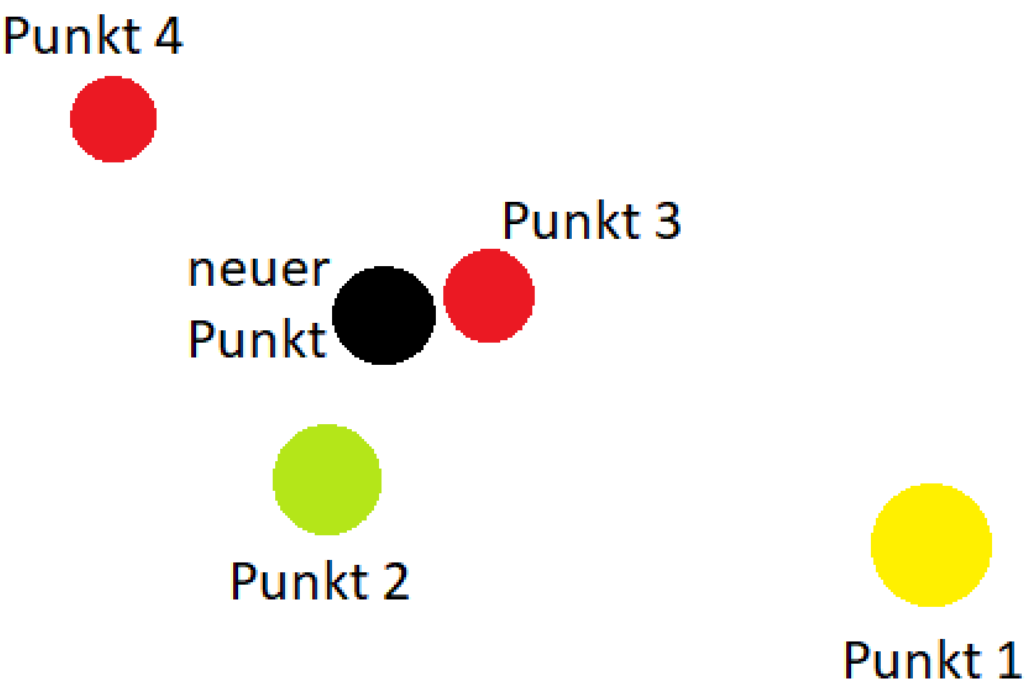
https://www.esslinger-zeitung.de/media/media.2181c3bc-5761-482c-9a2a-fa15a51b9dbb.16x9_1024.jpg

5. Ein Bild einem Cluster zuordnen (k-nächste-Nachbarn)

Der k-nächste-Nachbarn-Algorithmus ist ein Algorithmus zum Zuordnen eines neuen Wertes in ein bereits bestehendes Cluster.

“k” steht für die Anzahl der Nachbarn eines neuen Wertes, welche betrachtet werden.

5.1 Gebe die Nummern der drei nächsten Nachbarn des neuen Punktes **an**.

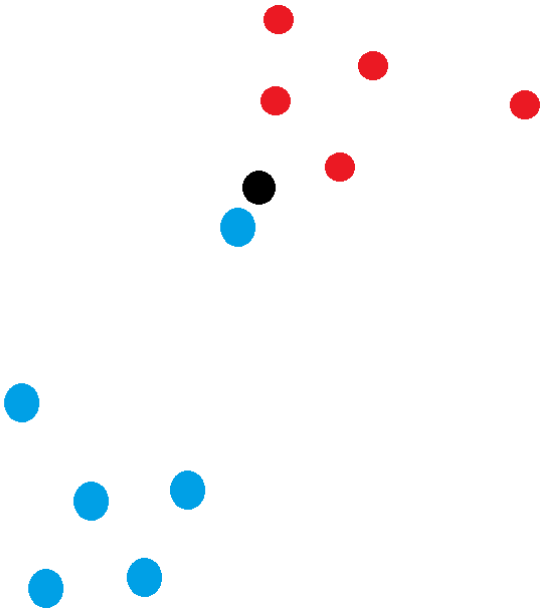
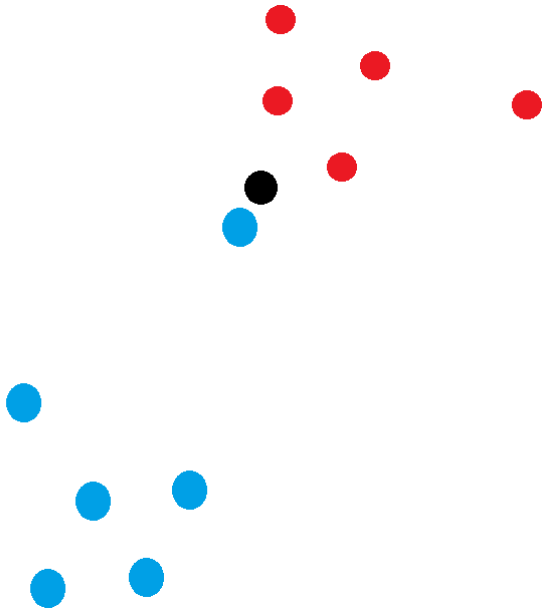
	
Nummer des nächsten Nachbars:	_____
Nummer des zweitnächsten Nachbarn:	_____
Nummer des drittnächsten Nachbarn:	_____

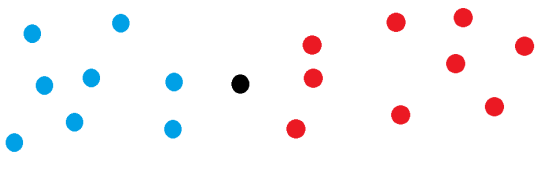
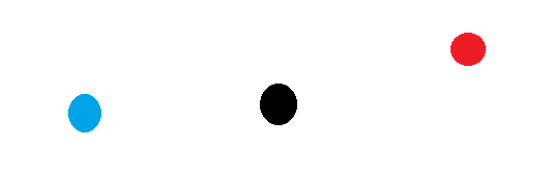
5.2 Basierend auf den drei nächsten Nachbarn: **Vermute** die Farbe des neuen Punktes, **begründe** deine Wahl kurz.

--

Wir betrachten jetzt die Angehörigkeit zu einem Cluster mithilfe von Farben.

5.3 Bestimme die Farbe der k nächsten Nachbarn des schwarzen Punktes, anschließend **bestimme** die dabei am häufigsten vorkommende Farbe und die entsprechende Zuordnung.
Hinweis: k (oben links) entspricht der Anzahl der Nachbarn.

<p>k=3</p> 	<p>k=1</p> 
Farben der Nachbarn: _____	Farben der Nachbarn: _____
Häufigste Farbe: _____	Häufigste Farbe: _____
Zugeordnete Farbe: _____	Zugeordnete Farbe: _____

<p>k=5</p> 	<p>k=2</p> 
Farben der Nachbarn: _____	Farben der Nachbarn: _____
Häufigste Farbe: _____	Häufigste Farbe: _____
Zugeordnete Farbe: _____	Zugeordnete Farbe: _____

5.4 Was ist dir aufgefallen in Bezug auf die Wahl eines Zahlenwertes k und die Eindeutigkeit der Zuordnung? **Beschreibe** kurz deine Gedanken.

5.5 Neben der Wahl des Zahlenwertes k , müssen wir als Entwickler auch entscheiden, was wir machen, wenn es mehrere häufigste Farben gibt. **Entwerfe** einen oder mehrere Vorschläge, wie man trotzdem eine Zuordnung treffen kann und **begründe** diese kurz.

Der k -nächste-Nachbarn-Algorithmus hat hier in ca. 9 von 10 Fällen korrekt funktioniert. Hingegen gibt es auch kompliziertere Verfahren, welche zum Beispiel auf Deep Learning basieren. Diese funktionieren zwar besser, hingegen funktioniert aufgrund der Variation der Gesichtsbilder, dem möglichen Daten Bias und der Ähnlichkeit von Gesichtern kein Verfahren immer perfekt. Zusätzlich wird der Erfolg der Verfahren eingeschränkt, durch die Vielzahl an Bedingungen, welche an eine Gesichtsdatenbank gestellt werden müssen, damit eine korrekte Zuordnung überhaupt erfolgen kann.

6. Clustering

Damit wir das Gesicht von Max Mustermann erkennen können, benötigen wir eine Datenbank, welche bereits Bilder von Max Mustermann enthält, sonst ist diese Person uns vollkommen unbekannt.

Die Frage ist: "Wie erstellen wir eine derartige Datenbank?"

Wir betrachten zwei Möglichkeiten:

- Wir füllen die Datenbank mit unbekannten Gesichtern und probieren herauszufinden, welche Bilder zusammengehören
- Wir füllen die Datenbank mit Gesichtern, von denen wir den Namen wissen

Für Fortgeschrittene (kann übersprungen werden):

Ein Beispiel, wie man zusammengehörende Bilder bestimmen kann, ist der gleich visualisierte Algorithmus, bekannt als k-means-Algorithmus. Dieser teilt eine Menge von Datenpunkten in eine vorgegebene Anzahl an Clustern auf. Dabei schätzt er die means (Mittelpunkte) der Cluster iterativ ab und ordnet die Datenpunkte über den Abstand den Clustern zu.

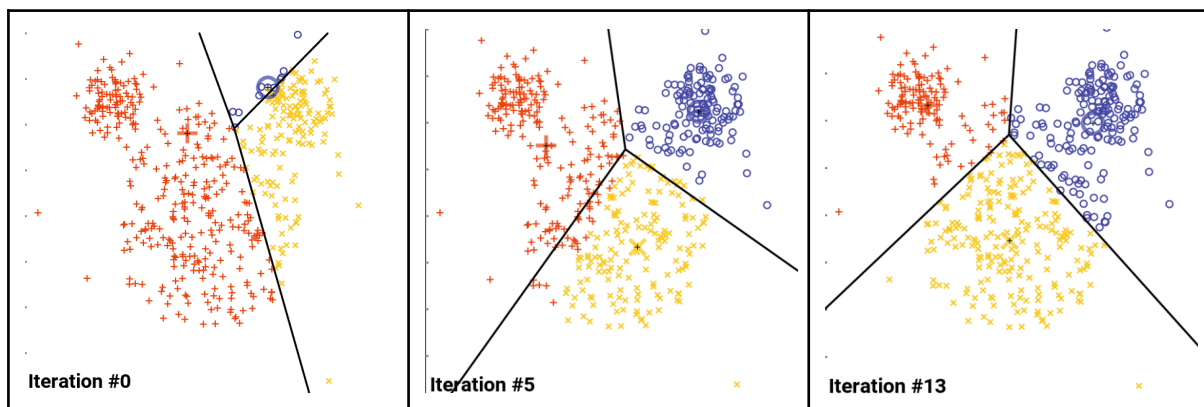


Abb. 11 k-means-Algorithmus (https://en.wikipedia.org/wiki/File:K-means_convergence.gif)

Pseudocode:

Eingabe:

Datenpunkte $X = \{x_1, x_2, \dots, x_n\}$

Anzahl Cluster k

Ausgabe:

Cluster-Zuordnungen der Datenpunkte

means M

Algorithmus:

Wähle zufällig k Datenpunkte aus X als means $M = \{m_1, \dots, m_k\}$.

Wiederhole, bis die Abbruchbedingung erfüllt ist:

-ordne jeden Datenpunkt x_i dem nächstgelegenen means m_j zu

-berechne für jedes nun entstandene cluster j den neuen means m_j aus

-Abbruchbedingung: means bleiben unverändert

Ausgabe:

k viele Cluster von Datenpunkten

k viele means der Cluster

Weiterlesen: <https://de.wikipedia.org/wiki/K-Means-Algorithmus>

Für Fortgeschrittene (kann übersprungen werden):

Beantworte kurz die folgende Frage. Warum ist die Abbruchbedingung, dass die means unverändert bleiben, eine sinnvolle Abbruchbedingung?

Der k-means-Algorithmus wurde betrachtet, da er in seiner Vorgehensweise dem k-nächste-Nachbarn-Algorithmus ähnelt. Im Vergleich zu anderen Clustering-Verfahren liefert er auf hochdimensionalen Daten, wie etwa den Wertevektoren von Bildern, oft gute Ergebnisse. Eine Einschränkung besteht jedoch darin, dass die Anzahl der zu bildenden Cluster (k) im Voraus gewählt werden muss. Eine falsche Wahl kann das Ergebnis negativ beeinflussen.

Für alle (nicht überspringen):

Angenommen, wir füllen eine Datenbank mit einer großen Menge von Gesichtern, zu denen wir bereits die zugehörigen Namen wissen.

Wir kennen jetzt bereits den k-nächste-Nachbarn-Algorithmus, welche verwendet werden kann, um ein neues Gesicht einem Cluster/einer Person zuzuordnen.

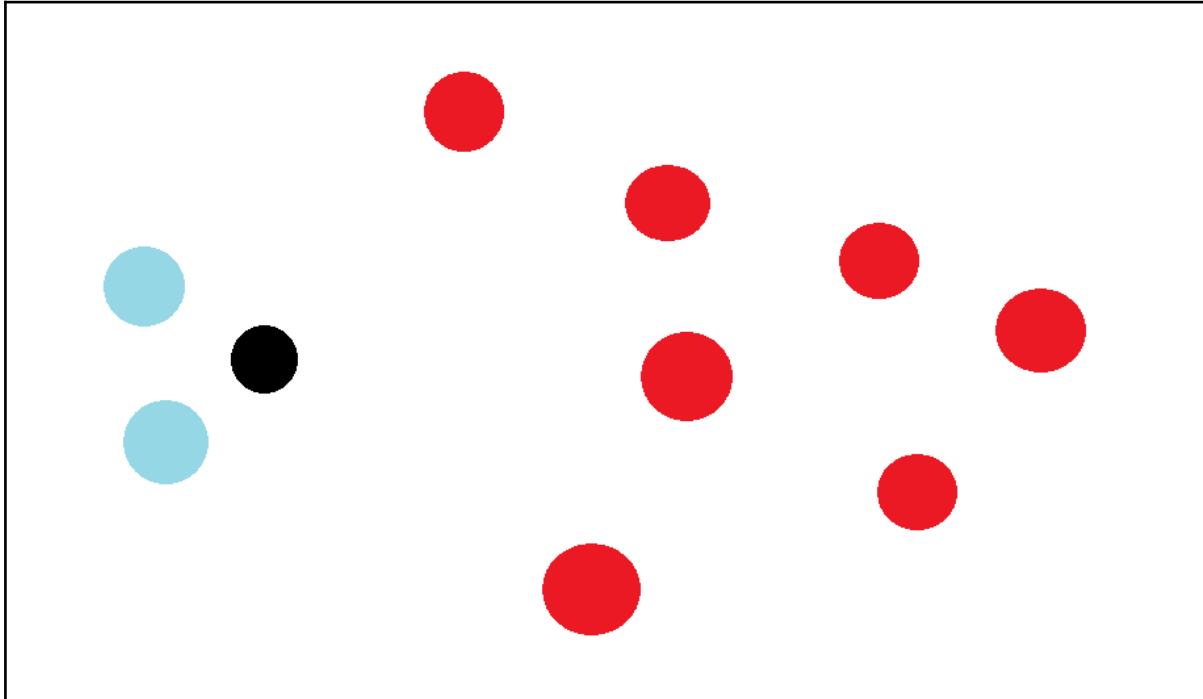
6.1 Was passiert, wenn ein Gesicht einer Person zugeordnet werden soll, welche in der Datenbank nicht existiert? **Beantworte** diese Frage kurz.

Hinweis: Eine Zuordnung findet immer statt.

6.2 Was kann passieren, wenn zwei nah aneinanderliegende Cluster (von Gesichtern) sich in ihrer Menge an Bildern stark unterscheiden? **Beantworte** diese Frage kurz mithilfe des folgenden Bildes.

Hinweis: Betrachte die Zuordnung mit $k=5$ für den k-nächste-Nachbarn-Algorithmus.

(Welchem Cluster wird der Punkt zugeordnet und welchem sollte er zugeordnet werden?)



Antwort:

6.3 Nenne ein paar Eigenschaften, welche eine Datenbank erfüllen sollte, damit diese besser geeignet ist zur Gesichtserkennung. **Begründe** diese kurz.

Hinweis: Eigenschaften basierend auf 6.1 und 6.2 reichen aus, trotzdem können gerne mehr Vermutungen aufgestellt werden.

(nach Aufgabe 4.2 stehen auch zwei mögliche Bedingungen)

7. Implementieren (k-nächste-Nachbarn | Zusatz)

(Das folgende Kapitel kann freiwillig bearbeitet werden und falls du vorzeitig fertig bist mit dem Arbeitsheft, kann dieses im Anschluss noch bearbeitet werden.)

Nun implementieren wir einen vereinfachten Klassifikator für den k-nächste-Nachbarn-Algorithmus in Python. Die folgenden Methoden sollen in der Datei *implementieren.py* eingefügt werden. Wir betrachten zweidimensionale Punkte, wobei andere Datenformate ähnlich implementiert werden können. Bei grundlegenden Problemen mit dem Bearbeiten und Durchführen von Code, bitte an die Lehrkraft wenden.

Datenformat:

Die Daten, welche verwendet werden und an dem der Algorithmus getestet werden soll, haben die folgende Form:

Bereits kategorisierte Punkte:

```
daten=[      [ [ x-Koordinate , y-Koordinate ] , Kennzeichnung ],
               [ [ x-Koordinate , y-Koordinate ] , Kennzeichnung ],
               ... ,
               [ [ x-Koordinate , y-Koordinate ] , Kennzeichnung ] ]
```

somit sind:

```
daten[i]:           Werte des i-ten Punktes
daten[i][0]:        Koordinaten des i-ten Punktes
daten[i][1]:        Kennzeichnung des i-ten Punktes
daten[i][0][0]:     x-Koordinate des i-ten Punktes
daten[i][0][1]:     y-Koordinate des i-ten Punktes
```

Neuer einzufügender Punkt:

```
neu=[x-Koordinate, y-Koordinate]
```

somit sind:

```
neu[0]:             x-Koordinate des neuen Punktes
neu[1]:             y-Koordinate des neuen Punktes
```

Anzahl des Nachbarn:

k=eine natürliche Zahl größer gleich 1

Punkte:

```
P=[x-Koordinate, Koordinate]
```

Abstand:

Um bewerten zu können, welche die nächsten Nachbarn zu einem Punkt sind, wird eine Funktion zum Bestimmen des Abstands zwischen zwei Punkten benötigt.

Der euklidische Abstand d zwischen zwei Punkten $P_1(x_1, y_1), P_2(x_2, y_2)$ ist definiert durch:

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

(der euklidische Abstand)

7.1 Vervollständige den Code der Methoden “abstand()” in der Datei *implementieren.py*, welche als Eingabe zwei Punkte wie oben beschrieben erhält und als Ausgabe den euklidischen Abstand ausgibt.

```
#Eingabe: Punkt 1 P1, Punkt 2 P2
#Ausgabe: euklidischer Abstand d
def abstand(P1,P2):

    #TODO

    return d
```

Abb. 12: leerer Quellcode von “abstand()”

Nun wollen wir den k-nächste-Nachbarn-Algorithmus implementieren. Dieser soll Werte “neu”, “Daten” und “k” wie oben als Eingabe erhalten und als Ausgabe die zugeordnete Kennzeichnung des neuen Punktes ausgeben.

Hinweise:

- Der neue Punkt “neu” muss nicht in die Datenliste “daten” eingefügt werden.
- Die Liste “daten_sorted” ist bereits vorgegeben, diese ist eine Kopie von “daten”, welche bereits sortiert ist nach dem Abstand zu “neu”.

7.2 Vervollständige die Methode “knn()” in *implementieren.py*.

Hinweis: Es müssen nur noch:

1. die k-nächsten Nachbarn mithilfe der sortierten Liste bestimmt werden
2. die häufigste Kennzeichnung der k-nächsten Nachbarn bestimmt werden

```
def knn(daten, neu, k):  
    daten_sorted=sort_by_abstand(daten, neu)  
  
    #TODO  
  
    return kennzeichnung
```

Abb. 13: leerer Quellcode von “knn()”

Hinweis:

Die Methode “test()” kann verwendet werden, um zu überprüfen, ob die Ausgaben von “knn()” mit den erwarteten Ausgaben übereinstimmen.

7.3 (Zusatzaufgabe):

(Kann gerne jetzt oder später bearbeitet werden.)

In Aufgabe 5.5 haben wir bereits kurz darüber nachgedacht, was wir machen, wenn es einen Gleichstand gibt, bei der häufigsten Kennzeichnung der Nachbarn.

Implementiere deinen Lösungsansatz oder **erstelle** passenden Pseudocode für deine Lösungsidee.

Pseudocode:

8. Anwendung an echten Bilddaten

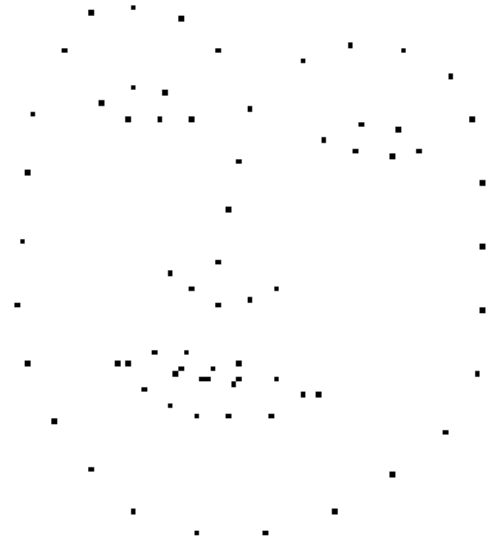


Abb. 14 Bild und Gatter (LFW Dataset (2007))

Wir sind mit den technischen Grundlagen nun fertig und haben auch bereits einige Fehlerquellen kennengelernt, die dafür sorgen können, dass beim Prozess der Gesichtserkennung etwas schief läuft.

Jetzt testen wir einige dieser Dinge an echten Bilddaten und werden sehen, dass diese größtenteils funktionieren, aber nicht immer.

Im Folgenden soll die Python Datei **SuSBefehle.py** verwendet und verschiedene Methoden aufgerufen werden. Mit der Methode **help()** erhältst du eine kleine Übersicht an Befehlen, welche du gerne auch selbständig verwenden kannst.

Bei den meisten Gesichtern hat das Erstellen der Wertevektoren problemlos funktioniert, aber es gab auch ein paar Fälle, wo es fehlgeschlagen ist.

Betrachte mithilfe von `Zeig_Bild_und_Gatter(Person_Nr, Bild_Nr)` in `SuSBefehle.py` die entsprechenden Bilder und **benenne** jeweils eine mögliche Fehlerursache.

Person Nummer	Bildnummer	mögliche Fehlerursache
0	63	
1	60	
2	35	
3	219	
5	52	

(Diese Aufgabe kann jetzt oder später bearbeitet werden.)

[illegible]

Ein Bild einem Cluster zuordnen

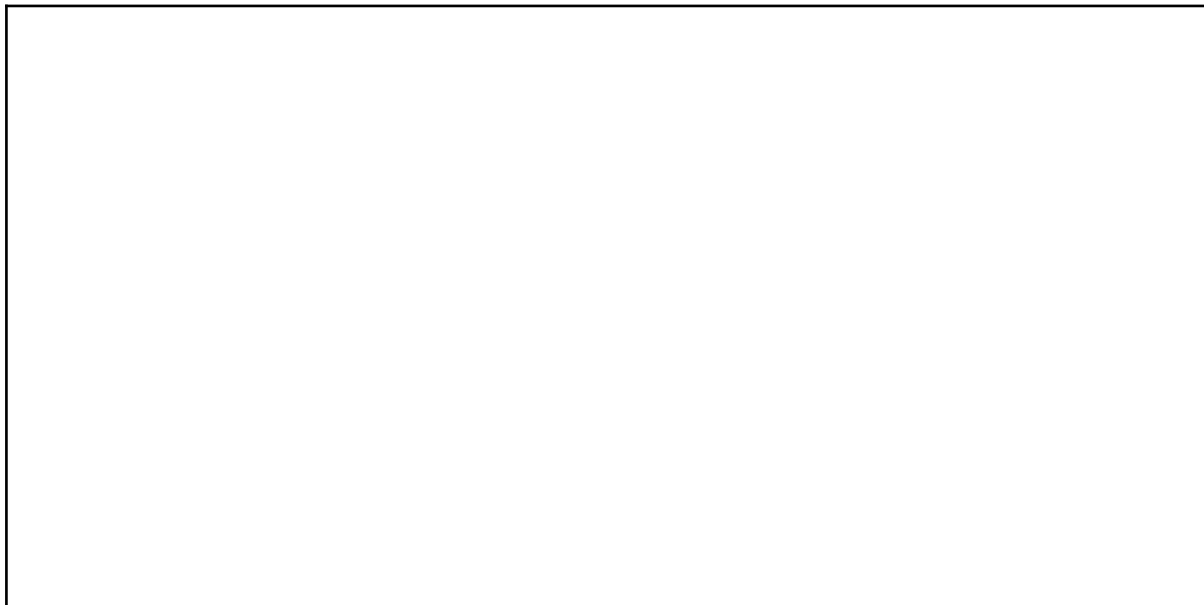
Daher, dass in der LFW-Gesichtsdatenbank jedes Gesicht bereits einer Person zugeordnet ist, betrachten wir jetzt nicht das Erstellen von Clustern, sondern wir nehmen einfach an, dass diese bereits existieren.

Von jeder der acht Personen befinden sich hier 40 Bilder in der Datenbank. Es wird von jeder Person ein Bild genommen, was nicht in diesen 40 Bildern ist und mithilfe des k-nächste-Nachbarn-Algorithmus wird eine Zuordnung durchgeführt

8.3 Öffnen Sie die Datei *SuSBefehle.py* und verwenden Sie den Befehl `test(k)`. Dieser verwendet den k-nächste-Nachbarn-Algorithmus für ein frei wählbares k .

Beschreibe, was du bei verschiedenen Werten für k beobachten konntest. Wie gut funktioniert der Algorithmus?

Hinweis: Achte darauf, bei welchen Werten für k der Algorithmus sehr gut oder schlecht funktioniert.



Einer der Hauptgründe dafür, dass der Algorithmus hier nur teilweise gut funktioniert, ist die Wahl der Abstandsfunktion. Der Abstand zwischen den Gattern wurde mithilfe des euklidischen Abstands (wie in Aufgabe 6.1) zwischen jedem Punkt paarweise bestimmt. Eine andere Abstandsfunktion würde die Trefferquote erhöhen, trotzdem würden Fehlzusordnungen passieren. Einer der Hauptgründe dafür ist, dass der verwendete Datensatz verhältnismäßig klein ist und wenig Variation enthält.

9. gesellschaftliche Reflexion

Gesichtserkennung wird in vielen Bereichen verwendet. Zum Beispiel:

- zur Entsperrung von Smartphones
- an Grenzen und Flughäfen
- in der Polizeiüberwachung
- im Marketing
- bei der automatischen Fotoorganisation

Trotz hoher Genauigkeit sind viele Systeme nicht neutral, sie spiegeln die Daten wider, mit denen sie trainiert wurden.

Die folgenden Schritte beschreiben den generellen Ablauf einer Gesichtserkennung.

Betrachten wir nochmal, wie der bisher kennengelernte Prozess abläuft.

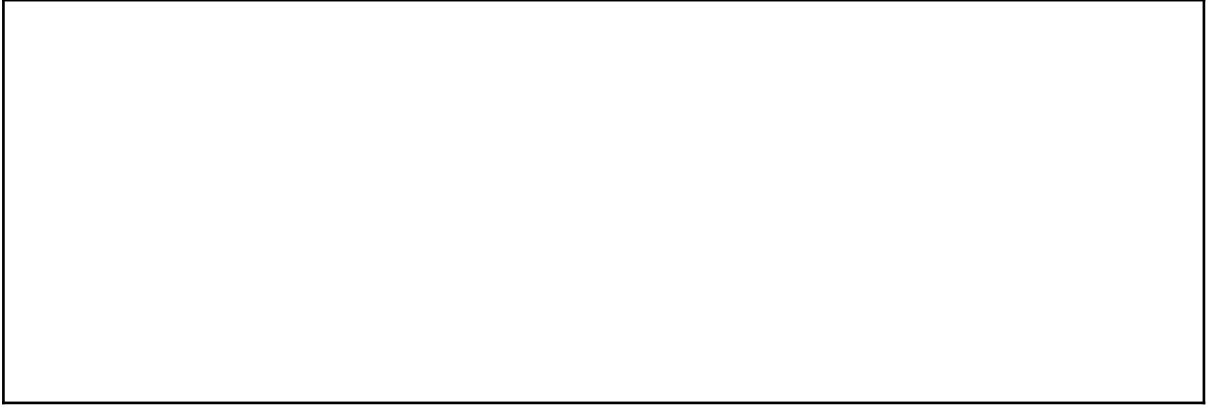
9.1 (Wiederholung) Nummeriere die folgenden Schritte mithilfe von Zahlen, um diese in eine sinnvolle Reihenfolge zu bringen.

Nummer	Schritt	Beschreibung
	Vergleich mit bekannten Gesichtern	Abgleich mit einer Datenbank
	Erkennung des Gesichts	Position vom Gesicht im Bild bestimmen
	Erfassen eines Bildes	Bilder erhalten z.B. über Kamera
	Ergebnis ausgeben	Wurde die Person erkannt/zugeordnet?
	Merkmale extrahieren	Form des Gesichts, Augenabstand, ...

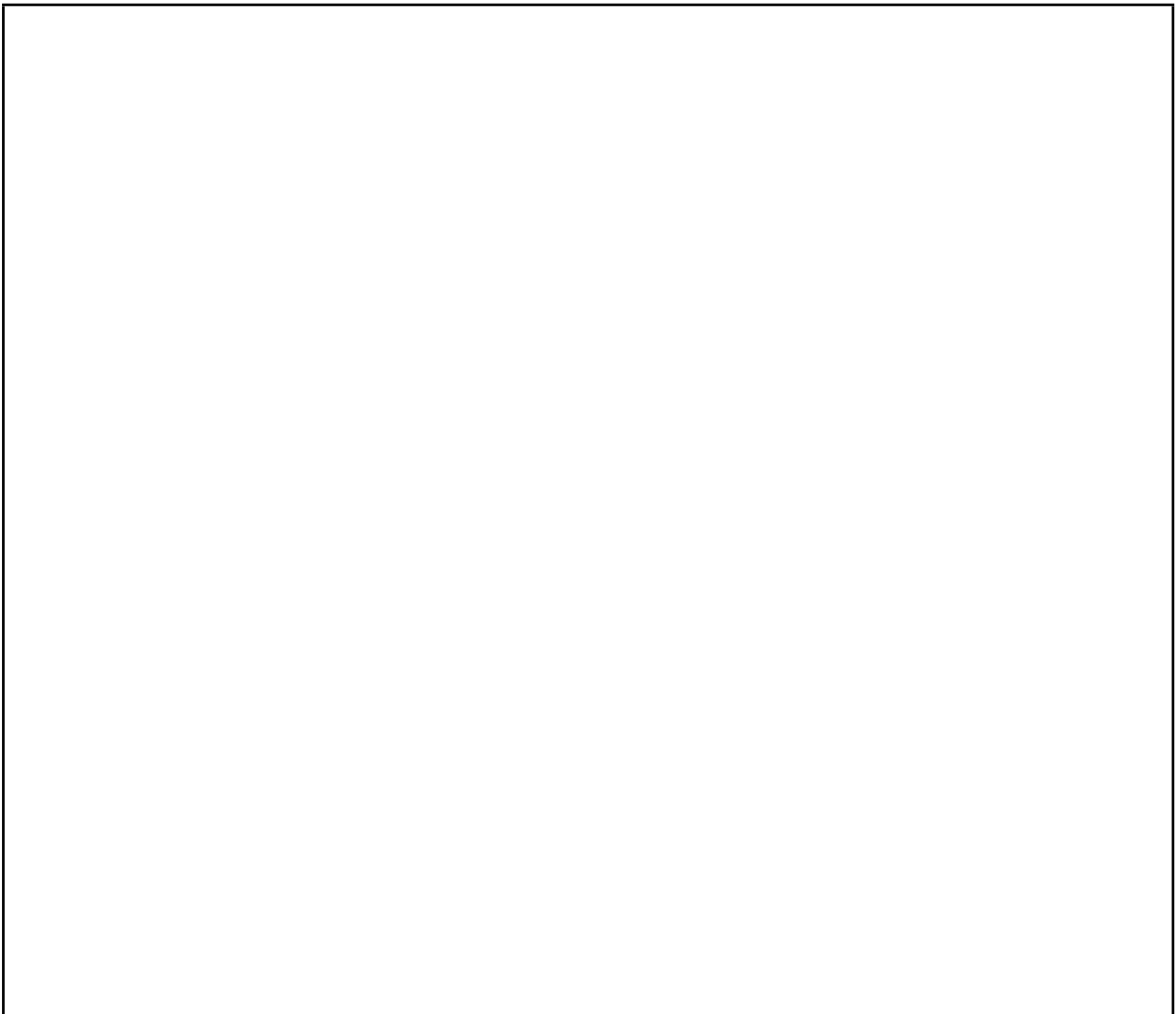
Wir haben bereits einige Fehlerquellen der Gesichtserkennung bestimmt.

9.2 (Wiederholung) Nenne mindestens drei Ursachen, durch die etwas auf technischer Ebene schiefgehen könnte bei der Gesichtserkennung.

Hinweis: Einige haben wir bisher schon kennengelernt.

A large, empty rectangular box with a thin black border, intended for the student to write their answer to question 9.2.

9.3 Erkläre, warum Gesichtserkennungssysteme auf große Datenbanken mit Gesichtsdaten angewiesen sind.

A large, empty rectangular box with a thin black border, intended for the student to write their answer to question 9.3.

Material A (Infotext):

In einigen Ländern werden bereits große nationale Datenbanken aufgebaut, welche biometrische Daten wie Gesichter, Fingerabdrücke und Augen enthalten.

Befürworter argumentieren, dass dadurch Kriminalität schneller aufgeklärt und Terroranschläge verhindert werden können. Auch im Alltag ist die Technik praktisch, zum Beispiel zum Entsperren von Handys oder Ermitteln von Anwesenheit.

Kritiker warnen jedoch vor einem Überwachungsstaat. Wenn jede Person auf Straßen, in Bahnhöfen, auf Flughäfen oder beim Einkaufen ständig überwacht wird, leidet das Recht auf Privatsphäre. Zusätzlich könnten Daten gestohlen oder missbraucht werden.

Ein weiteres Problem ist, dass Gesichtserkennungssysteme nicht fehlerfrei sind. Teilweise zeigen Studien, dass manche Gruppen (z.B. aufgrund von Ethnie oder Geschlecht) häufiger falsch erkannt werden. Dies kann zu Diskriminierung führen.

Material B (Tabelle):

Chancen	Risiken
Mehr Sicherheit in öffentlichen Räumen	Verlust der Privatsphäre
Schnelleres Auffinden gesuchter Personen	Gefahr auf Überwachungsstaat
Komfort (Handy, Passkontrolle)	Datenmissbrauch durch Fremde oder Behörden
Personalisierung	Diskriminierung durch fehlerhafte Erkennung

9.4 Vergleiche mithilfe der Materialien (A und B) die Chancen und Risiken der Gesichtserkennung

9.5 Beurteile, ob der Nutzen von Gesichtserkennung die Risiken überwiegt. Beziehe dich dabei auch das Problem möglicher Diskriminierung. Beziehe dich ebenfalls auf technische Schwächen der Gesichtserkennungssysteme.

Nun bist du am Ende des Arbeitsheftes angekommen, falls andere Schülerinnen und Schüler fertig sind, könnt ihr gemeinsam zu der folgenden Frage **diskutieren**.

“Wo setzt ihr die Grenze? In welchen Bereichen würdet ihr Gesichtserkennung befürworten und in welchen Bereichen seht ihr die Verwendung kritisch?”

Sonst kannst du jetzt Zusatzaufgaben, welche du übersprungen hast, bearbeiten oder du kannst selbständig in den folgenden Textquellen dich weiter vertiefen.

Viel Erfolg!



Textquellen zur Vertiefung:

fachlich:

Automatische Gesichtserkennung: Methoden und Anwendungen:

http://www.medien.ifi.lmu.de/fileadmin/mimuc/hs_ws0506/papers/Automatische_Gesichtserkennung.pdf

gesellschaftlich:

Gesichtserkennung – Technologie zwischen Unterstützung und Unterdrückung:

https://www.fotomuseum.ch/wp-content/uploads/2023/12/Gesichtserkennung_DE.pdf

rechtlich:

Gesichtserkennung – Ein Diskussionsbeitrag zur Regulierung der Technologie

https://www.bidt.digital/wp-content/uploads/sites/2/2022/08/bidt-Reihe-Impulse-3-Gesichtserkennung_Online.pdf