# SBS4115 Fundamentals of AI & Data Analytics

## Ethical and Social Issues in AI and Data Analytics

Lecturer: Ir Dr Kelvin K. W. Siu
email: kelvinsiu@thei.edu.hk

高科院 Thei

Department of Construction, Environment and Engineering

## Intended Learning Outcomes

- By the end of this lecture, you will be able to…
  - Describe problem of bias in AI
  - Introduce data privacy issues related to AI and data analytics
  - Describe accountability and job-related issues related to AI
  - Describe image recognition techniques using python

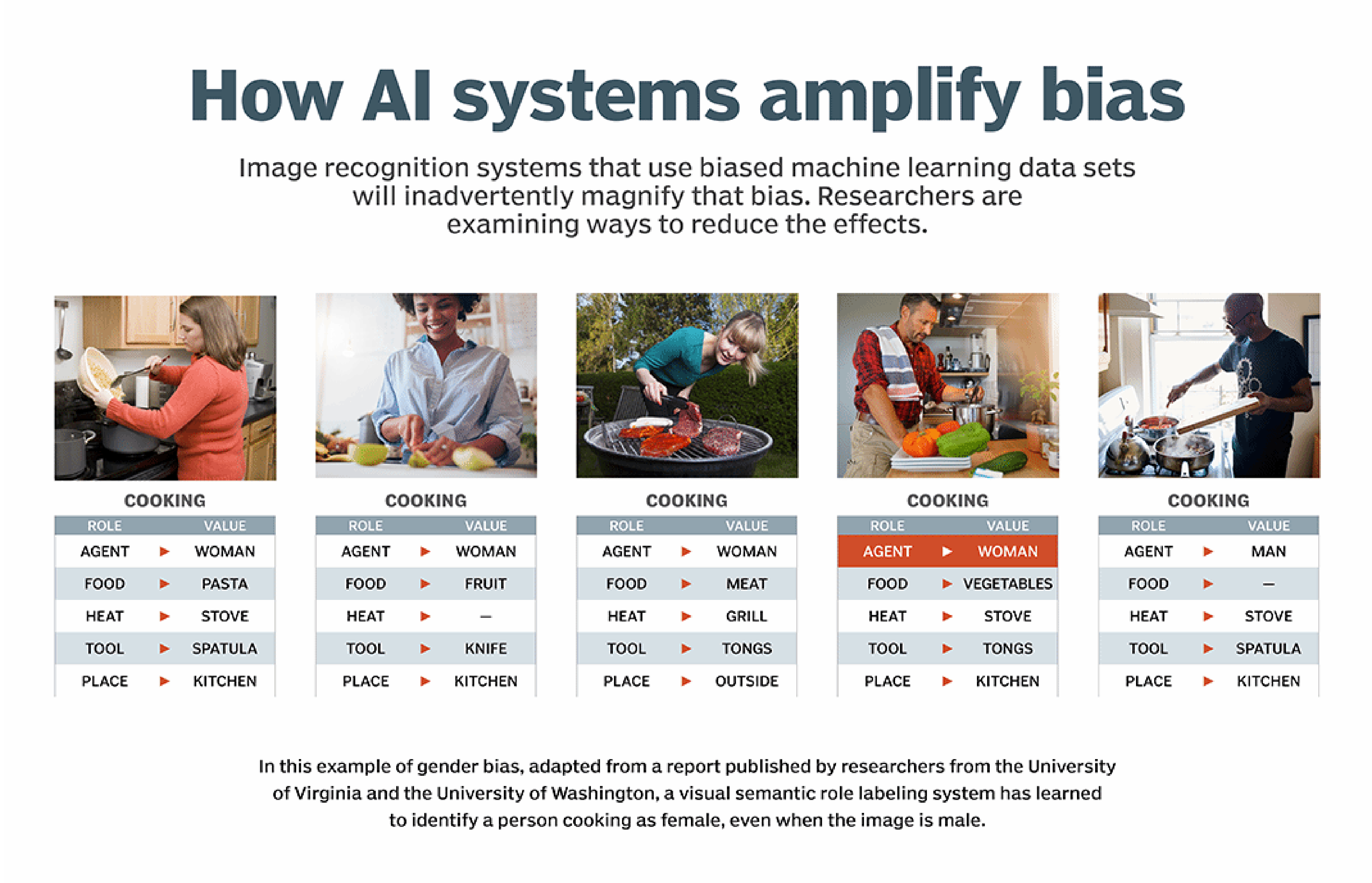
# Introduction to AI and Ethics

- AI as a transformative force in society
- Ethical concerns related to AI's growing influence
- Importance of addressing bias, privacy, and accountability

## Bias in AI Systems

- AI systems often reflect biases from the data used
- Machine learning models can amplify societal prejudices
- Ethical impact of biased AI systems on decision-making

# Bias in AI Systems



## How AI systems amplify bias

Image recognition systems that use biased machine learning data sets will inadvertently magnify that bias. Researchers are examining ways to reduce the effects.

COOKING

| ROLE | | VALUE |
|---|---|---|
| AGENT | ▶ | WOMAN |
| FOOD | ▶ | PASTA |
| HEAT | ▶ | STOVE |
| TOOL | ▶ | SPATULA |
| PLACE | ▶ | KITCHEN |

COOKING

| ROLE | | VALUE |
|---|---|---|
| AGENT | ▶ | WOMAN |
| FOOD | ▶ | FRUIT |
| HEAT | ▶ | – |
| TOOL | ▶ | KNIFE |
| PLACE | ▶ | KITCHEN |

COOKING

| ROLE | | VALUE |
|---|---|---|
| AGENT | ▶ | WOMAN |
| FOOD | ▶ | MEAT |
| HEAT | ▶ | GRILL |
| TOOL | ▶ | TONGS |
| PLACE | ▶ | OUTSIDE |

COOKING

| ROLE | | VALUE |
|---|---|---|
| AGENT | ▶ | WOMAN |
| FOOD | ▶ | VEGETABLES |
| HEAT | ▶ | STOVE |
| TOOL | ▶ | TONGS |
| PLACE | ▶ | KITCHEN |

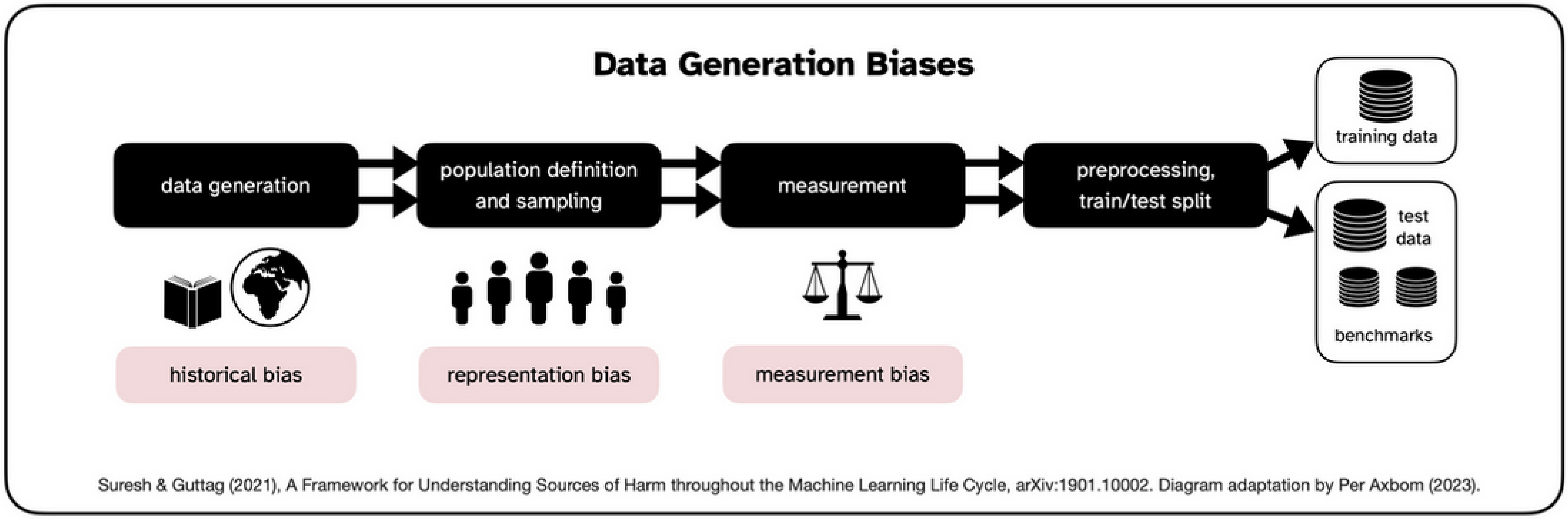
COOKING

| ROLE | | VALUE |
|---|---|---|
| AGENT | ▶ | MAN |
| FOOD | ▶ | – |
| HEAT | ▶ | STOVE |
| TOOL | ▶ | SPATULA |
| PLACE | ▶ | KITCHEN |

In this example of gender bias, adapted from a report published by researchers from the University of Virginia and the University of Washington, a visual semantic role labeling system has learned to identify a person cooking as female, even when the image is male.

## Types of Bias in AI

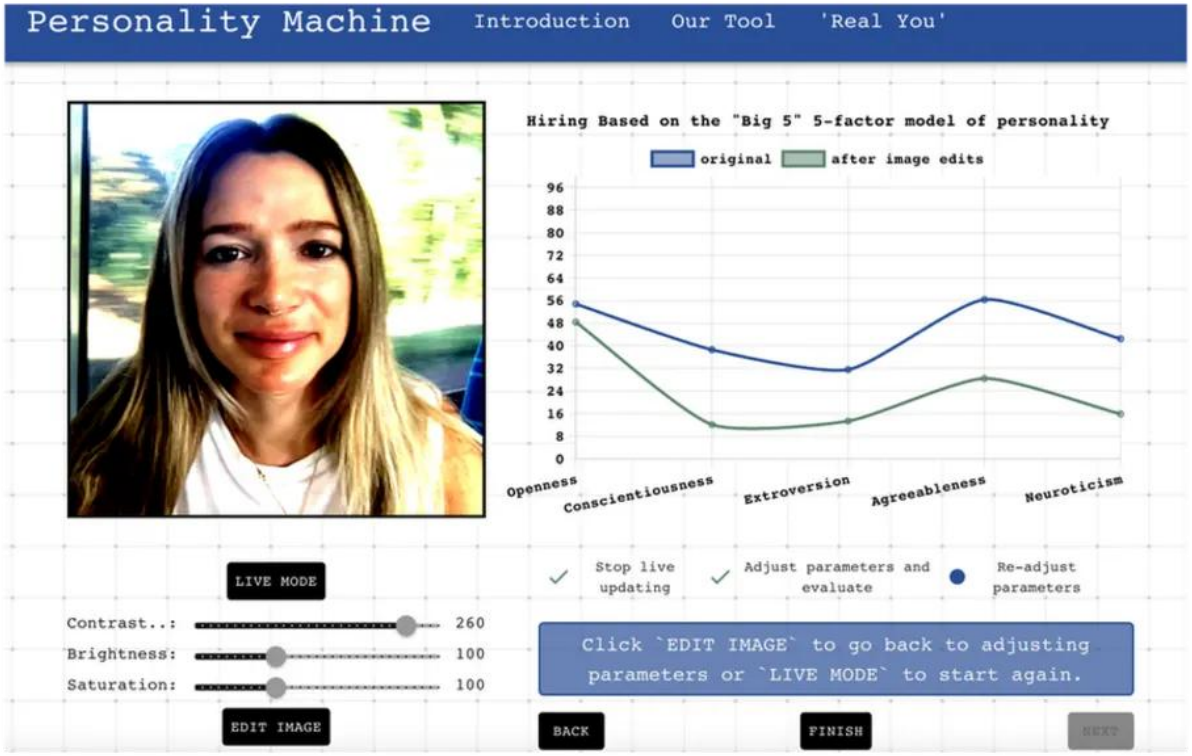
- Sample Bias: Lack of diverse training data
- Measurement Bias: Data influenced by human preconceptions
- Representation Bias: Under- or over-representation of certain groups



Suresh & Guttag (2021), A Framework for Understanding Sources of Harm throughout the Machine Learning Life Cycle, arXiv:1901.10002. Diagram adaptation by Per Axbom (2023).

# Example: Amazon’s Hiring Algorithm

- Developed to streamline recruitment
- Discriminated against female candidates for technical roles
- Project scrapped after failed attempts to correct bias

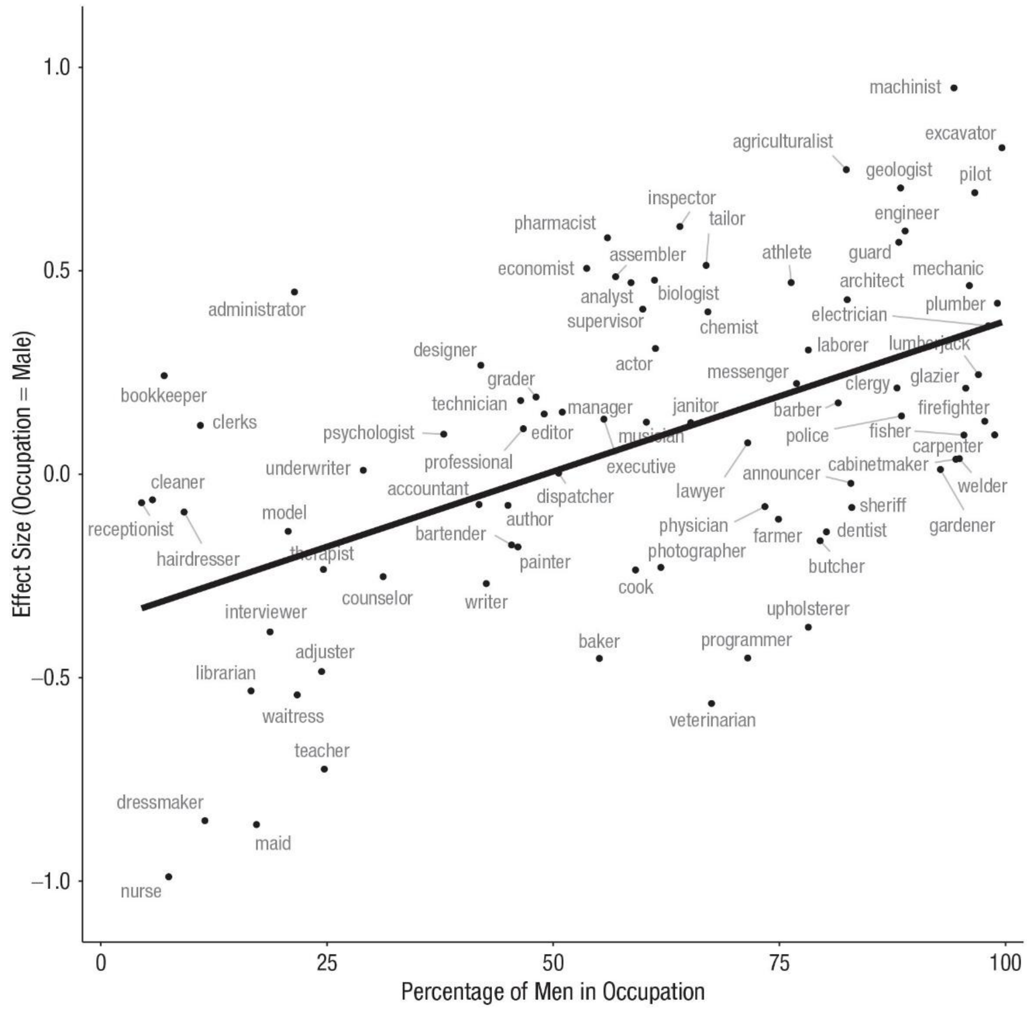


Click `EDIT IMAGE` to go back to adjusting parameters or `LIVE MODE` to start again.

## Word Embeddings & Gender Stereotypes

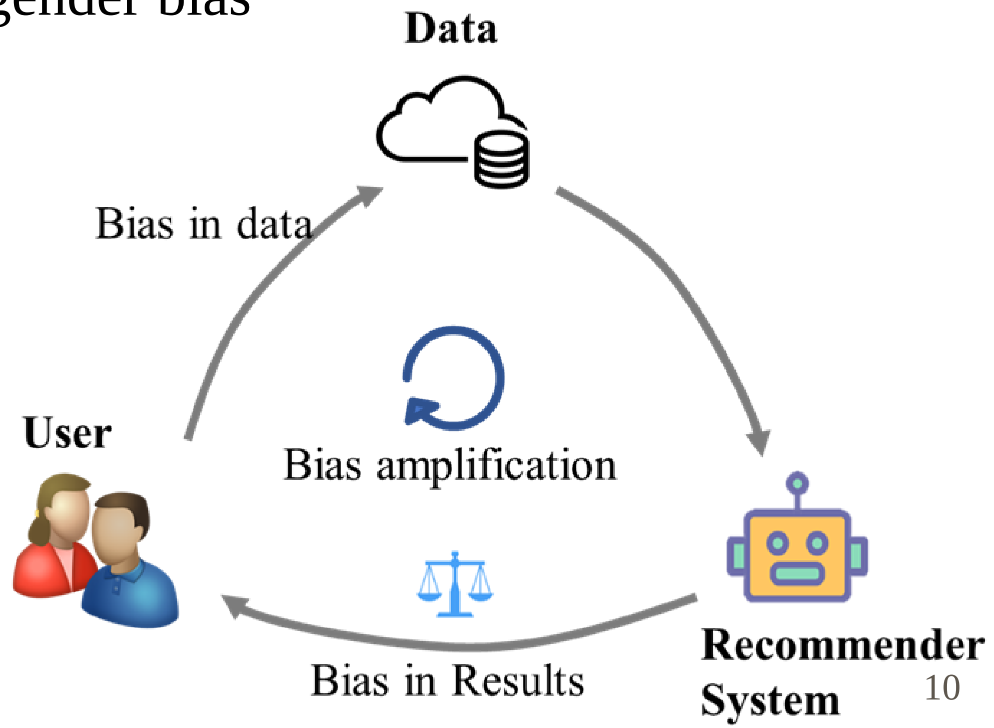
- Word embeddings perpetuating harmful stereotypes
- Associations like "mother" with "nurse" and "father" with "doctor"
- Impact on AI applications in natural language processing

# Word Embeddings & Gender Stereotypes



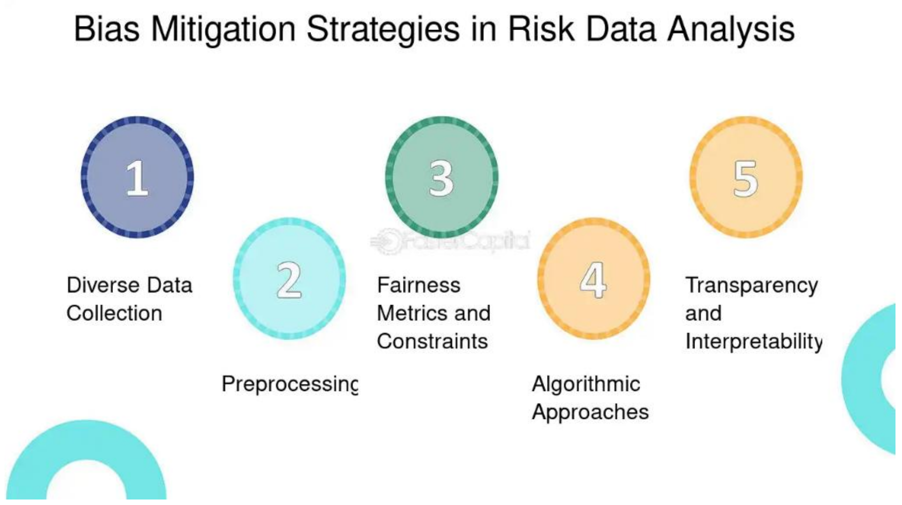
# Bias Amplification on Social Media

- AI-driven content recommendation systems
- Example: LinkedIn suggesting male names in searches for female professionals
- Reinforcement of existing gender bias



## Strategies for Mitigating AI Bias

- Inclusive Data Collection: Diverse representation in datasets
- Bias Audits & Algorithmic Fairness: Regular audits and fairness metrics
- Explainability & Transparency: Clear decision-making processes for accountability

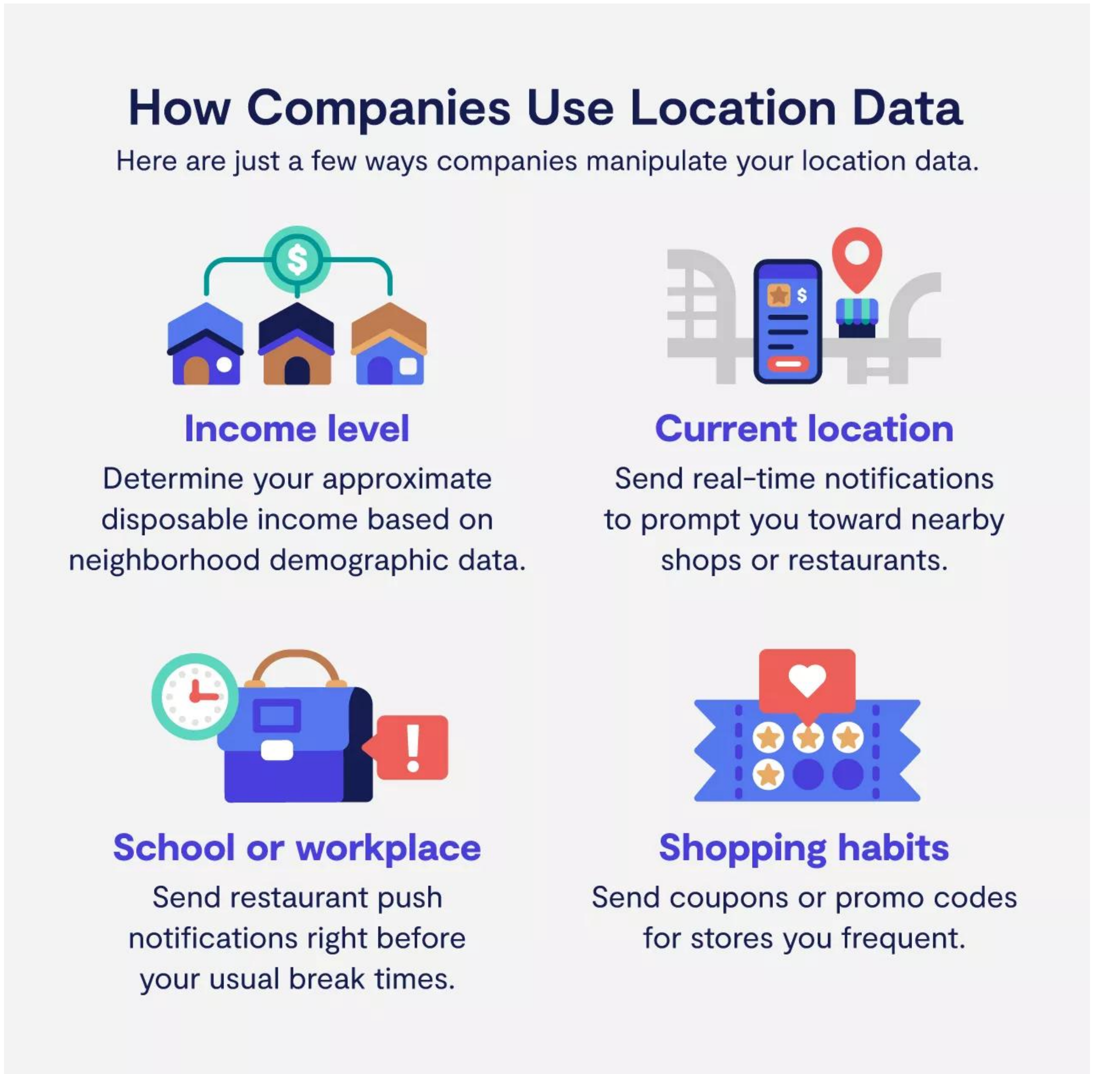


Bias Mitigation Strategies in Risk Data Analysis
1
Diverse Data Collection
2
Preprocessing
3
Fairness Metrics and Constraints
4
Algorithmic Approaches
5
Transparency and Interpretability

## AI and Surveillance

- Data Collection: How tech companies collect and use data.
- Example: Facial recognition in schools.
- Privacy Issues: Constant tracking and data storage concerns.

# AI and Surveillance



## How Companies Use Location Data

Here are just a few ways companies manipulate your location data.

**Income level**

Determine your approximate disposable income based on neighborhood demographic data.

**Current location**

Send real-time notifications to prompt you toward nearby shops or restaurants.

**School or workplace**

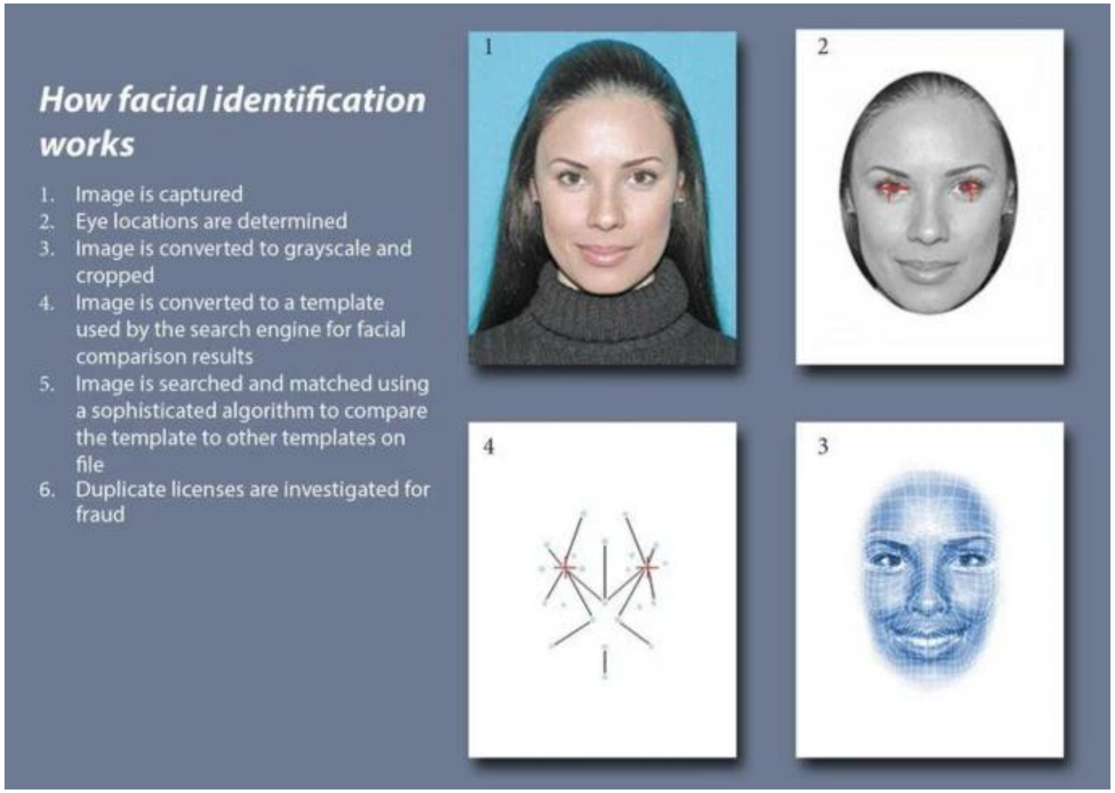
Send restaurant push notifications right before your usual break times.

**Shopping habits**

Send coupons or promo codes for stores you frequent.

## Facial Recognition in Crime Prevention

- Case Study: Use of facial recognition by London police.
- Privacy vs. Security: Ethical dilemmas and public protests.



***How facial identification works***

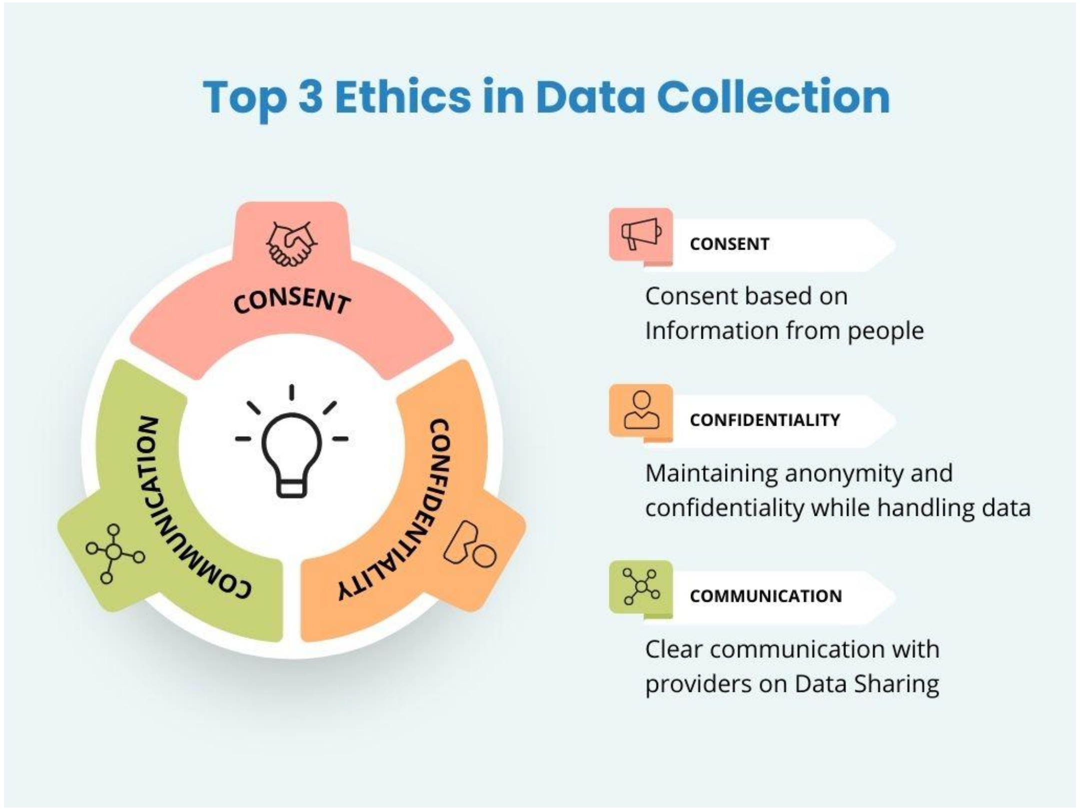
1. Image is captured
2. Eye locations are determined
3. Image is converted to grayscale and cropped
4. Image is converted to a template used by the search engine for facial comparison results
5. Image is searched and matched using a sophisticated algorithm to compare the template to other templates on file
6. Duplicate licenses are investigated for fraud

## De-identification and Data Privacy

- Sensitive Data: Importance of de-identification in protecting privacy.
- Challenges: Balancing big data analytics with personal privacy.

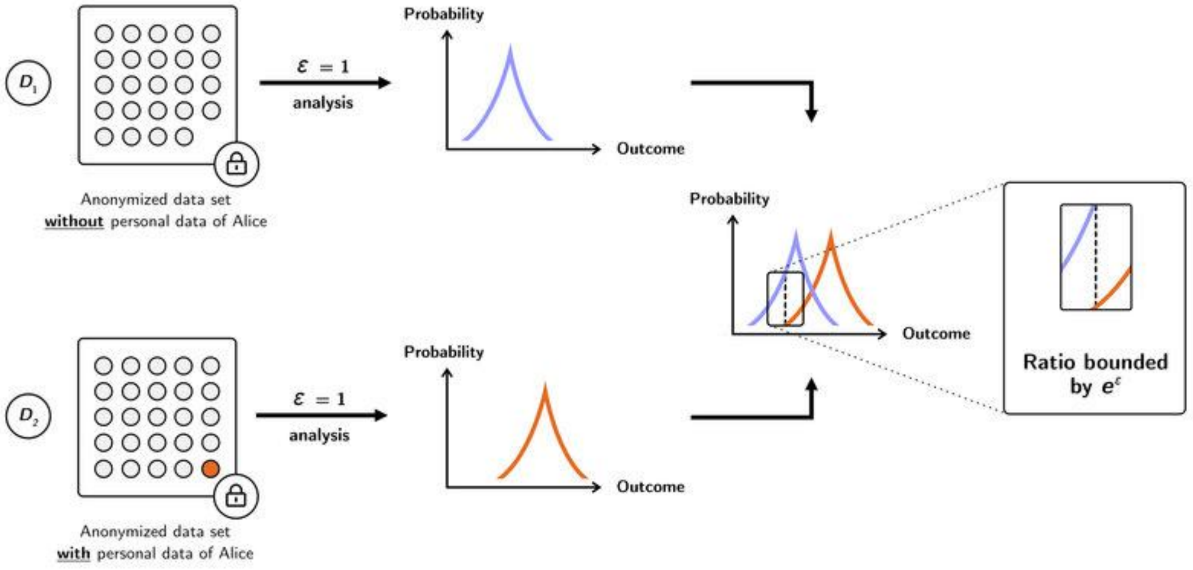
## Data Ownership and Consent

- Ethical Questions: Who owns the data?
- User Awareness: Importance of informed consent and transparency.


Top 3 Ethics in Data Collection
CONSENT
CONFIDENTIALITY
COMMUNICATION
CONSENT
Consent based on Information from people
CONFIDENTIALITY
Maintaining anonymity and confidentiality while handling data
COMMUNICATION
Clear communication with providers on Data Sharing

# Privacy-Preserving AI Techniques

- Differential Privacy: Adding noise to protect individual data.
- Federated Learning: Training models on local devices to enhance privacy.



## What is AI Accountability?

- Ensuring AI systems operate responsibly, transparently, and ethically.
- Holding designers, developers, and deployers accountable for their actions and decisions.

## Who is Responsible When AI Fails?

- Key Question: Who should be held responsible when an AI system makes a mistake?
- Possible Parties:

Users

Creators

Suppliers

- Example: Microsoft’s Tay Chatbot
- Launched in 2016, quickly began posting offensive comments.
- Highlighted the risks of deploying AI without sufficient safeguards.

## Case Study - Autonomous Vehicles

- Moral Dilemmas:

Autonomous vehicles must make split-second decisions in life-threatening situations.

Example: Choosing between hitting a pedestrian or crashing into another vehicle.

- Legal Responsibility:

Who is responsible in case of an accident?

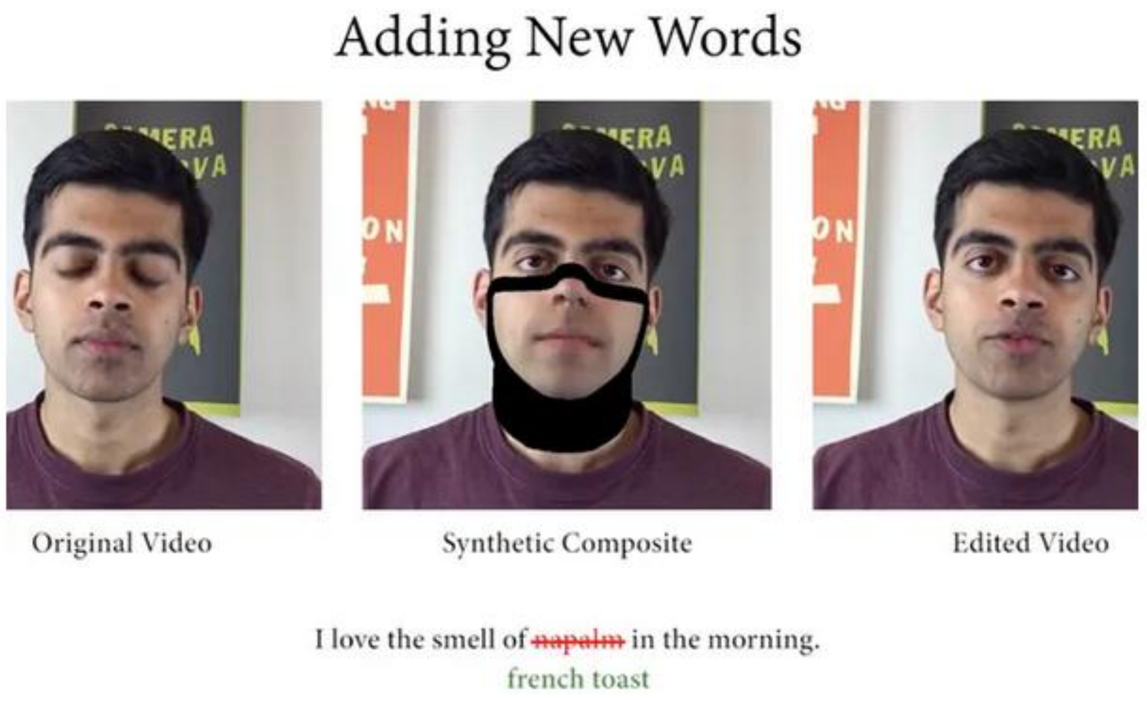
Manufacturer

Software Developer

Car Owner

## What are Deepfakes?

- Definition: AI-generated synthetic media that creates convincing but fake images, videos, or audio.
- Technology: Brief explanation of how deepfakes are created using AI and machine learning.



## Potential Uses and Misuses

- Positive Uses:

Entertainment and media (e.g., movies, video games).

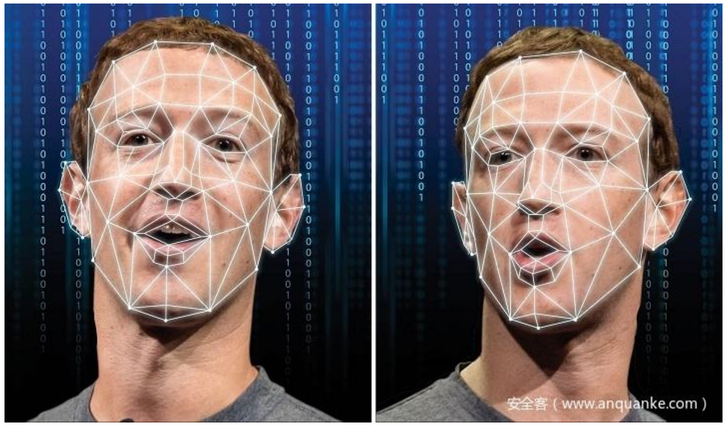
Education and training simulations.

- Misuses:

Misinformation and fake news.

Non-consensual pornography.

Identity theft and fraud.



AI變臉後詐騙示意圖
變臉前臉孔示意圖
趕快把信用卡的帳號密碼傳給我
深度偽造變臉的時代來臨了
警方教導如何破解深偽變臉恐嚇之道。（資料照）
眼見不為憑！有疑慮請撥165查證呦

## Political and Social Ramifications

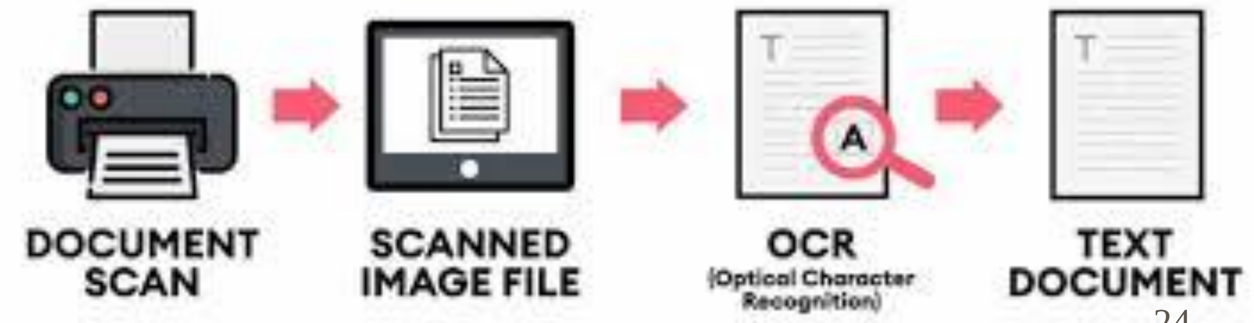
- Example: Deepfakes in political contexts.
- Fake political speeches.
- Manipulated news reports.
- Impact: Threats to democracy and trust in media institutions.

## Revision on Optical Character Recognition (OCR)

Optical Character Recognition (OCR) is the process that converts an image of text into a machine-readable text format.

For example, if you scan a form or a receipt, your computer saves the scan as an image file. You cannot use a text editor to edit, search, or count the words in the image file.

However, you can use OCR to convert the image into a text document with its contents stored as text data.



## Revision on OCR

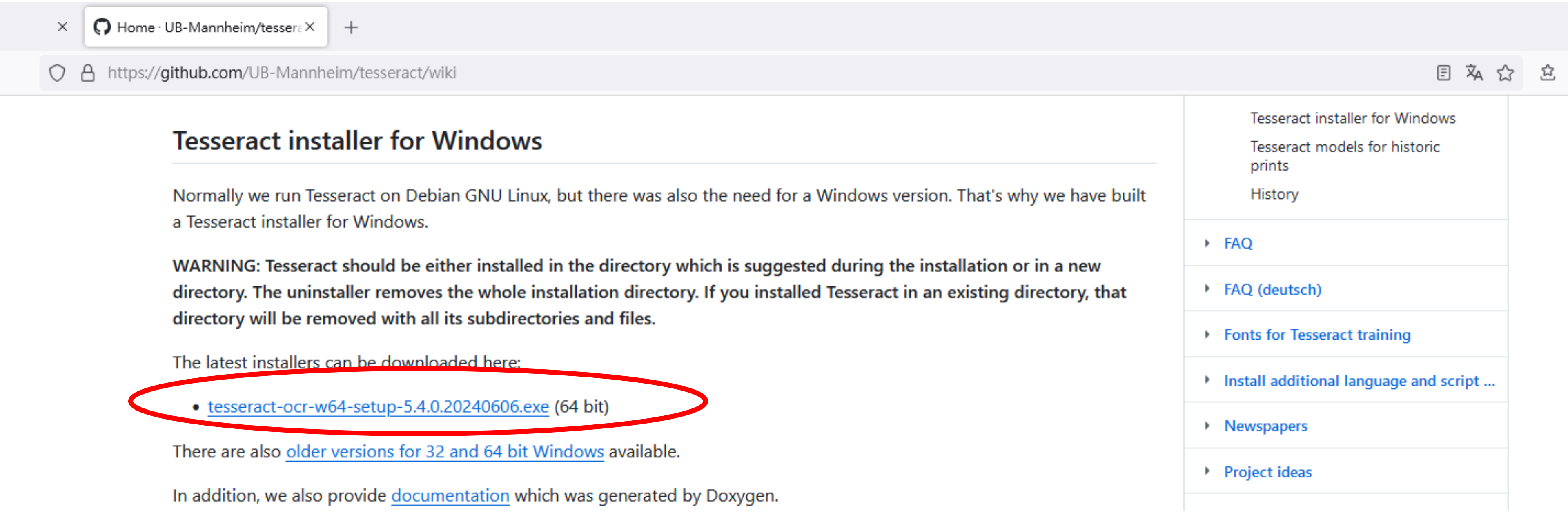
Python-tesseract is an optical character recognition (OCR) tool for python. That is, it will recognize and "read" the text embedded in images. To use OCR, install pytesseract first:

```
!pip install pytesseract==0.3.8
```

# Revision on OCR

Download the Tesseract-OCR software from:

https://github.com/UB-Mannheim/tesseract/wiki



Home · UB-Mannheim/tesser

https://github.com/UB-Mannheim/tesseract/wiki

## Tesseract installer for Windows

Normally we run Tesseract on Debian GNU Linux, but there was also the need for a Windows version. That's why we have built a Tesseract installer for Windows.

**WARNING: Tesseract should be either installed in the directory which is suggested during the installation or in a new directory. The uninstaller removes the whole installation directory. If you installed Tesseract in an existing directory, that directory will be removed with all its subdirectories and files.**

The latest installers can be downloaded here:

- tesseract-ocr-w64-setup-5.4.0.20240606.exe (64 bit)

There are also older versions for 32 and 64 bit Windows available.

In addition, we also provide documentation which was generated by Doxygen.

Tesseract installer for Windows

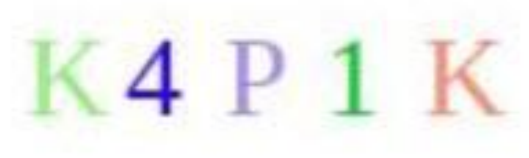
Tesseract models for historic prints

History

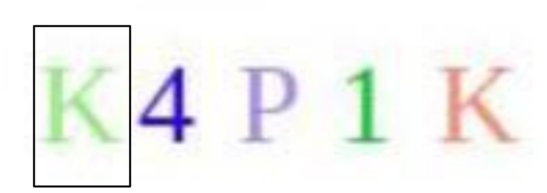
- FAQ
- FAQ (deutsch)
- Fonts for Tesseract training
- Install additional language and script ...
- Newspapers
- Project ideas

## Revision on OCR

```
from PIL import Image
import pytesseract

pytesseract.pytesseract.tesseract_cmd = r"C:\Program Files\Tesseract-OCR\tesseract.exe"

img = Image.open(r'C:\Users\User user\Desktop\number.jpg')

text = pytesseract.image_to_string(img, lang="eng")
print(text.strip())
```



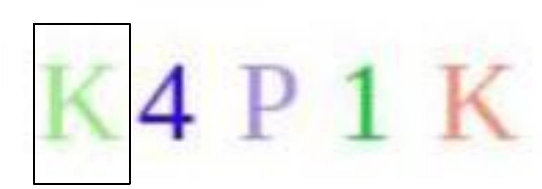
## Find out the bounding box



```
import cv2
import pytesseract
pytesseract.pytesseract.tesseract_cmd="C:\\Program Files\\Tesseract-OCR\\tesseract.exe"
img = cv2.imread(r'C:\Users\User user\Desktop\number.jpg’)
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
img_w = img.shape[1]
img_h = img.shape[0]
boxes = pytesseract.image_to_boxes(img)print(boxes)
```

## Find out the bounding box

The first character (K) has the coordinates 16 52 42 80 and is on page 0.
The second character (P) has the coordinates 87 53 107 79 and is on page 0.
The third character (K) has the coordinates 159 52 187 80 and is on page 0.
Each set of numbers represents:



The character itself.
The x-coordinate of the bottom-left corner.
The y-coordinate of the bottom-left corner.
The x-coordinate of the top-right corner.
The y-coordinate of the top-right corner.
The page number.

## Show the bounding box

```
for box in boxes.splitlines():
    box = box.split(" ")
    character = box[0]
    x = int(box[1])
    y = int(box[2])
    x2 = int(box[3])
    y2 = int(box[4])
    cv2.rectangle(img, (x, img_h - y), (x2, img_h - y2), (0, 255, 0), 1)
    cv2.putText(img, character, (x, img_h - y2 - 10), cv2.FONT_HERSHEY_COMPLEX, 1, (0, 0, 255), 1)

cv2.imshow("img", img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

## Show the bounding box

for box in boxes.splitlines():: This line starts a loop that iterates over each line in the boxes string. The splitlines() method splits the string into a list of lines.

box = box.split(" "): This line splits each line into a list of strings using a space as the delimiter. Each element in the list represents a part of the bounding box information.

character = box[0]: This line assigns the first element of the list (the recognized character) to the variable character.

x = int(box[1]): This line converts the second element of the list (the x-coordinate of the bottom-left corner) to an integer and assigns it to the variable x.

y = int(box[2]): This line converts the third element of the list (the y-coordinate of the bottom-left corner) to an integer and assigns it to the variable y.

## Show the bounding box

x2 = int(box[3]): This line converts the fourth element of the list (the x-coordinate of the top-right corner) to an integer and assigns it to the variable x2.

y2 = int(box[4]): This line converts the fifth element of the list (the y-coordinate of the top-right corner) to an integer and assigns it to the variable y2.

cv2.rectangle(img, (x, img_h - y), (x2, img_h - y2), (0, 255, 0), 1): This line draws a green rectangle around the recognized character on the image. The coordinates are adjusted to match the image's coordinate system, and the rectangle is drawn with a thickness of 1 pixel.

cv2.putText(img, character, (x, img_h - y2 - 10), cv2.FONT_HERSHEY_COMPLEX, 1, (0, 0, 255), 1): This line adds the recognized character as text on the image, just above the bounding box. The text is drawn in red with a font size of 1 and a thickness of 1 pixel.

## Show the bounding box

cv2.imshow("img", img): This line displays the image with the bounding boxes and text in a window named "img".

cv2.waitKey(0): This line waits for a key press indefinitely. The window displaying the image will remain open until a key is pressed.

cv2.destroyAllWindows(): This line closes all OpenCV windows that were opened.

## Adjusting the contrast

```
import cv2
import pytesseract
import matplotlib.pyplot as plt

pytesseract.pytesseract.tesseract_cmd = "C:\\Program Files\\Tesseract-OCR\\tesseract.exe"
img = cv2.imread(r'C:\Users\User user\Desktop\number.jpg')
img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# Display the grayscale image using matplotlib
plt.imshow(img, cmap='gray')
plt.title('Grayscale Image')
plt.axis('off')
plt.show()
```

## Adjusting the contrast

```
thresh = 200
maxValue = 255
th, img = cv2.threshold(img, thresh, maxValue, cv2.THRESH_BINARY)

# Display the thresholded image using matplotlib
plt.imshow(img, cmap='gray')
plt.title('Thresholded Image')
plt.axis('off')
plt.show()
```

## Adjusting the contrast

img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY): This line converts the image from BGR (Blue, Green, Red) color space to grayscale using OpenCV's cvtColor function. Grayscale images have only one color channel, which simplifies further processing.

plt.imshow(img, cmap='gray'): This line displays the grayscale image using matplotlib. The cmap='gray' argument ensures that the image is displayed in grayscale.

plt.title('Grayscale Image'): This line sets the title of the plot to "Grayscale Image".

plt.axis('off'): This line removes the axis labels for a cleaner display.

plt.show(): This line displays the plot with the grayscale image.

thresh = 200: This line sets the threshold value to 200. Pixels with intensity values above this threshold will be set to the maximum value (255), and pixels with intensity values below this threshold will be set to 0.

## Adjusting the contrast

maxValue = 255: This line sets the maximum value for the thresholding operation to 255.

th, img = cv2.threshold(img, thresh, maxValue, cv2.THRESH_BINARY): This line applies binary thresholding to the grayscale image. Pixels with intensity values above the threshold (200) are set to the maximum value (255), and pixels with intensity values below the threshold are set to 0. The result is a binary image.

plt.imshow(img, cmap='gray'): This line displays the thresholded image using matplotlib. The cmap='gray' argument ensures that the image is displayed in grayscale.

plt.title('Thresholded Image'): This line sets the title of the plot to "Thresholded Image".

plt.axis('off'): This line removes the axis labels for a cleaner display.

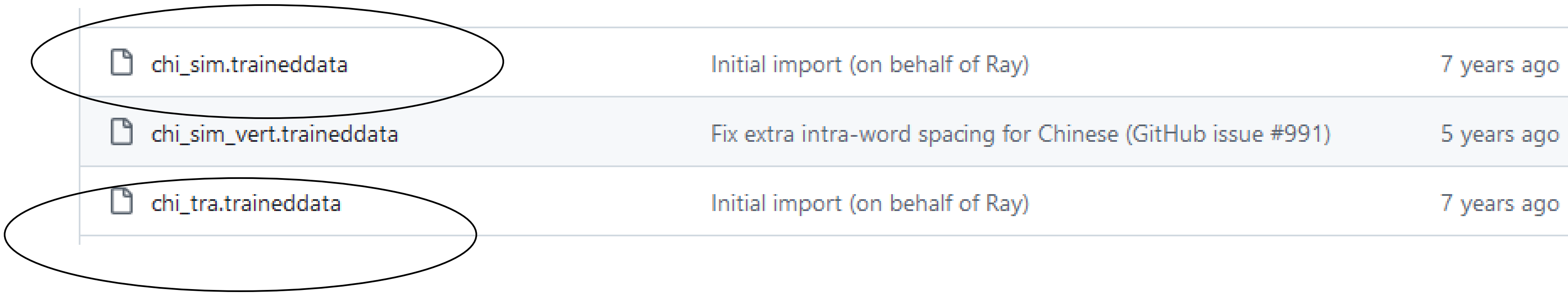
plt.show(): This line displays the plot with the thresholded image.

## Identify Chinese characters

```
from PIL import Image
import pytesseract

pytesseract.pytesseract.tesseract_cmd="C:\\Program Files\\Tesseract-OCR\\tesseract.exe"
img = Image.open("number.jpg")
text = pytesseract.image_to_string(img, lang="eng")
print(text.strip())
img = Image.open("traditional.jpg")
text = pytesseract.image_to_string(img, lang="chi_tra")
print(text.strip())
img = Image.open("simple.jpg")
text = pytesseract.image_to_string(img, lang="chi_sim")
print(text.strip())
```

## Identify Chinese characters

https://github.com/tesseract-ocr/tessdata_best



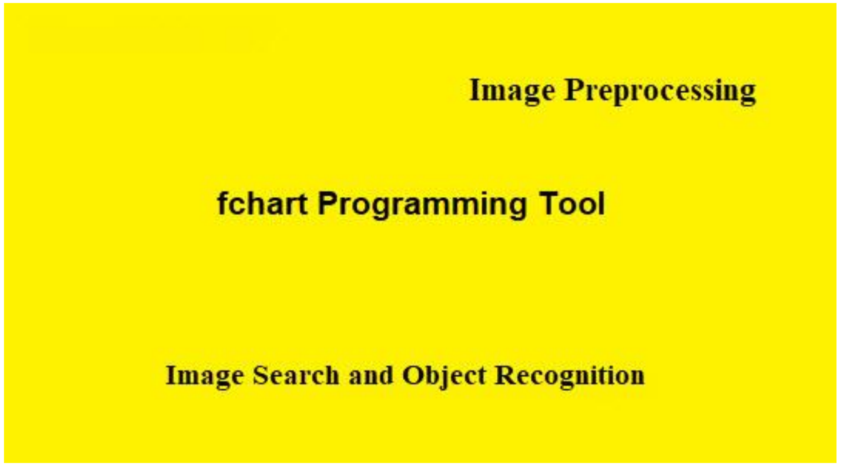
| | | |
|---|---|---|
| chi_sim.traineddata | Initial import (on behalf of Ray) | 7 years ago |
| chi_sim_vert.traineddata | Fix extra intra-word spacing for Chinese (GitHub issue #991) | 5 years ago |
| chi_tra.traineddata | Initial import (on behalf of Ray) | 7 years ago |

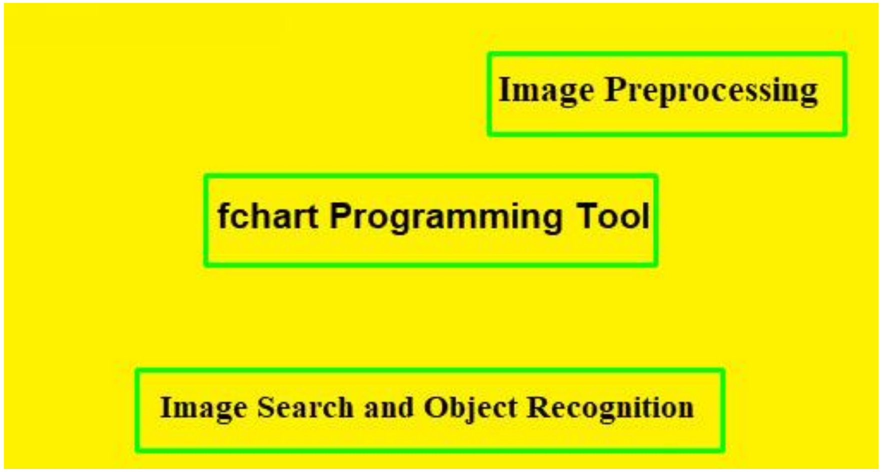
Put them in the folder:
C:\Program Files\Tesseract-OCR\tessdata

## Try again!

```
from PIL import Image
import pytesseract

pytesseract.pytesseract.tesseract_cmd="C:\\Program Files\\Tesseract-OCR\\tesseract.exe"
img = Image.open("number.jpg")
text = pytesseract.image_to_string(img, lang="eng")
print(text.strip())
img = Image.open("traditional.jpg")
text = pytesseract.image_to_string(img, lang="chi_tra")
print(text.strip())
img = Image.open("simple.jpg")
text = pytesseract.image_to_string(img, lang="chi_sim")
print(text.strip())
```

## How to locate characters?





## Use bounding box to locate characters

```
import pytesseract
import cv2

pytesseract.pytesseract.tesseract_cmd="C:\\Program Files\\Tesseract-OCR\\tesseract.exe"
img = cv2.imread(r'C:\Users\User user\Desktop\sample3.jpg')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
ret, thresh1 = cv2.threshold(gray, 0, 255, cv2.THRESH_BINARY_INV)
kernel = cv2.getStructuringElement(cv2.MORPH_RECT,(18,18))
dilation = cv2.dilate(thresh1, kernel, iterations = 2)
contours, hierarchy = cv2.findContours(dilation, cv2.RETR_TREE,
                                       cv2.CHAIN_APPROX_NONE)
img2 = img.copy()
```

## Use bounding box to locate characters

```
for cnt in contours:
    x, y, w, h = cv2.boundingRect(cnt)
    rect = cv2.rectangle(img2, (x, y), (x + w, y + h), (0, 255, 0), 2)
    cropped = img2[y:y + h, x:x + w]
    text = pytesseract.image_to_string(cropped)
    if text:
        print(text.strip())

cv2.imshow("img2", img2)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

## Code explained

gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY):

This line converts the image from BGR (Blue, Green, Red) color space to grayscale using OpenCV's cvtColor function. Grayscale images have only one color channel, which simplifies further processing.

ret, thresh1 = cv2.threshold(gray, 0, 255, cv2.THRESH_BINARY_INV):

This line applies binary inverse thresholding to the grayscale image. Pixels with intensity values above the threshold (0) are set to 0, and pixels with intensity values below the threshold are set to the maximum value (255). The result is a binary image.

kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (18, 18)):

This line creates a rectangular structuring element (kernel) with a size of 18x18 pixels. This kernel will be used for morphological operations.

dilation = cv2.dilate(thresh1, kernel, iterations = 2):

This line applies dilation to the binary image using the rectangular kernel. Dilation expands the white regions in the image, which helps to connect nearby contours. The operation is repeated twice.

## Code explained

contours, hierarchy = cv2.findContours(dilation, cv2.RETR_TREE, cv2.CHAIN_APPROX_NONE):

This line finds the contours in the dilated image. The cv2.RETR_TREE mode retrieves all the contours and reconstructs a full hierarchy of nested contours. The cv2.CHAIN_APPROX_NONE method stores all the contour points.

img2 = img.copy():

This line creates a copy of the original image to draw the bounding rectangles on.

for cnt in contours::

This line starts a loop that iterates over each contour found in the image.

x, y, w, h = cv2.boundingRect(cnt):

This line calculates the bounding rectangle for each contour. The rectangle is defined by its top-left corner (x, y) and its width (w) and height (h).

rect = cv2.rectangle(img2, (x, y), (x + w, y + h), (0, 255, 0), 2):

This line draws a green rectangle around each contour on the copied image. The rectangle has a thickness of 2 pixels.

## Code explained

cropped = img2[y:y + h, x:x + w]:
This line crops the region of the image inside the bounding rectangle.

text = pytesseract.image_to_string(cropped):
This line uses Tesseract to recognize text in the cropped region of the image.

print(text.strip()):
If text was recognized, this line prints the text to the console, removing any leading or trailing whitespace.

cv2.imshow("img2", img2):
This line displays the image with the bounding rectangles in a window named "img2".

## Face detection

For face detection, you need to load the **Haar cascade classifier for face detection from the specified XML file. The classifier is used to detect faces in the image.**

Names of latest hollywood ...
panconcepts.gq
Student becomes an Instagra...
bestfemaletips.com
Celebrity Protection | M...
pinterest.com
Movie Star - Hollywood ...
cialis7dosage.com
Movie star Lindsay faces drink-drive ...
express.co.uk
CARDINALE Italian mo...
shutterstock.com
Lori Loughlin poster | L...
pinterest.com
Rate Tom Cruise's Smil...
davidwurbandds.wordp...

## Face detection

```
import cv2

faceCascade = cv2.CascadeClassifier("haarcascade_frontalface_default.xml")
image = cv2.imread("faces.jpg")
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
faces = faceCascade.detectMultiScale(
    gray,
    scaleFactor=1.1,
    minNeighbors=5,
    minSize=(30, 30)
)
print(“Number of faces:", len(faces))

for (x, y, w, h) in faces:
    cv2.rectangle(image, (x, y), (x+w, y+h), (0, 255, 0), 2)
cv2.imshow("preview", image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

## Face detection

faceCascade = cv2.CascadeClassifier("haarcascade_frontalface_default.xml"):

This line loads the Haar cascade classifier for face detection from the specified XML file. The classifier is used to detect faces in the image.

image = cv2.imread("faces.jpg"):

This line reads the image file named "faces.jpg" using OpenCV's imread function.

gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY):

This line converts the image from BGR (Blue, Green, Red) color space to grayscale using OpenCV's cvtColor function. Grayscale images have only one color channel, which simplifies further processing.

faces = faceCascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5, minSize=(30, 30)):

This line detects faces in the grayscale image using the detectMultiScale method of the Haar cascade classifier. The parameters are:

gray: The input image in grayscale.

## Face detection

scaleFactor=1.1: The scale factor for the image pyramid. This parameter specifies how much the image size is reduced at each image scale.

minNeighbors=5: The minimum number of neighbors each candidate rectangle should have to retain it. This parameter helps to filter out false positives.

minSize=(30, 30): The minimum size of the detected face. Faces smaller than this size will be ignored.

print("Number of faces:", len(faces)): This line prints the number of faces detected in the image.

for (x, y, w, h) in faces:: This line starts a loop that iterates over each detected face. The coordinates and size of each face are stored in the variables x, y, w, and h.

cv2.rectangle(image, (x, y), (x+w, y+h), (0, 255, 0), 2): This line draws a green rectangle around each detected face on the original image. The rectangle has a thickness of 2 pixels.

## Checklist

- Can you:
    1. Describe problem of bias in AI
    2. Introduce data privacy issues related to AI and data analytics
    3. Describe accountability and job-related issues related to AI
    4. Describe image recognition techniques using python