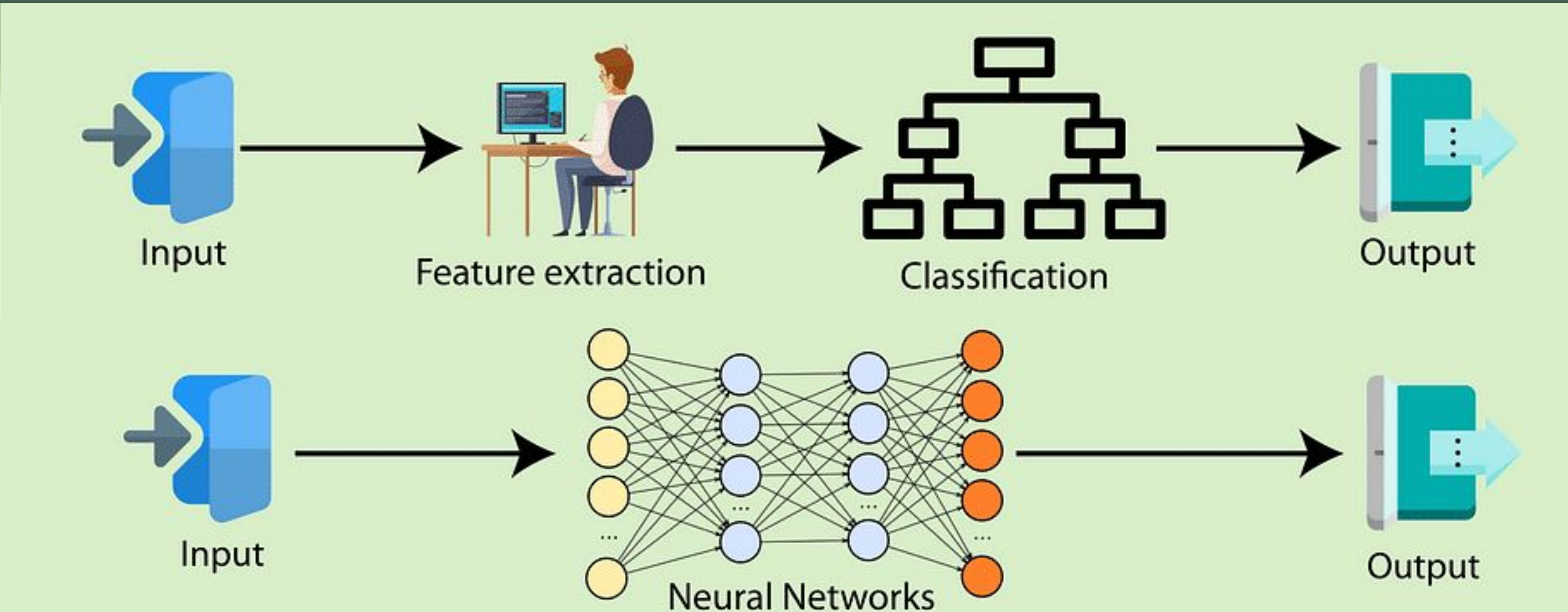
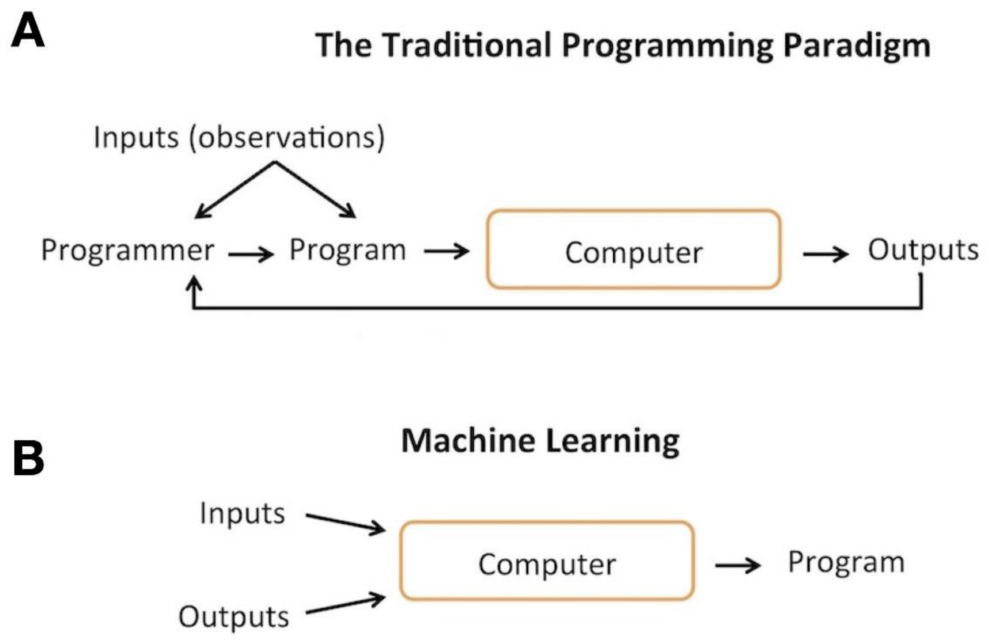
Introduction to Artificial Intelligence, Machine Learning and Deep Learning

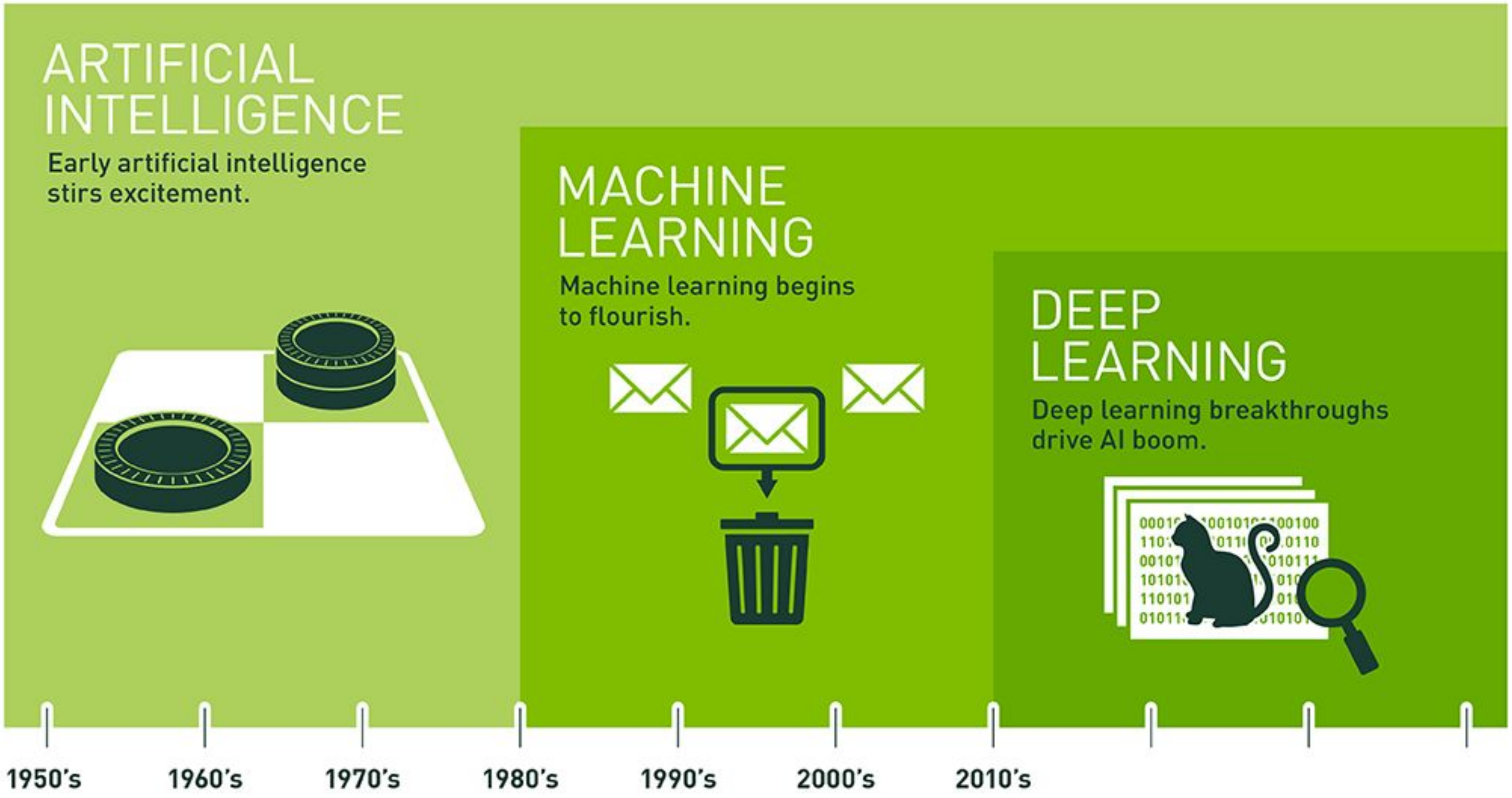


Computer Program and Machine Learning



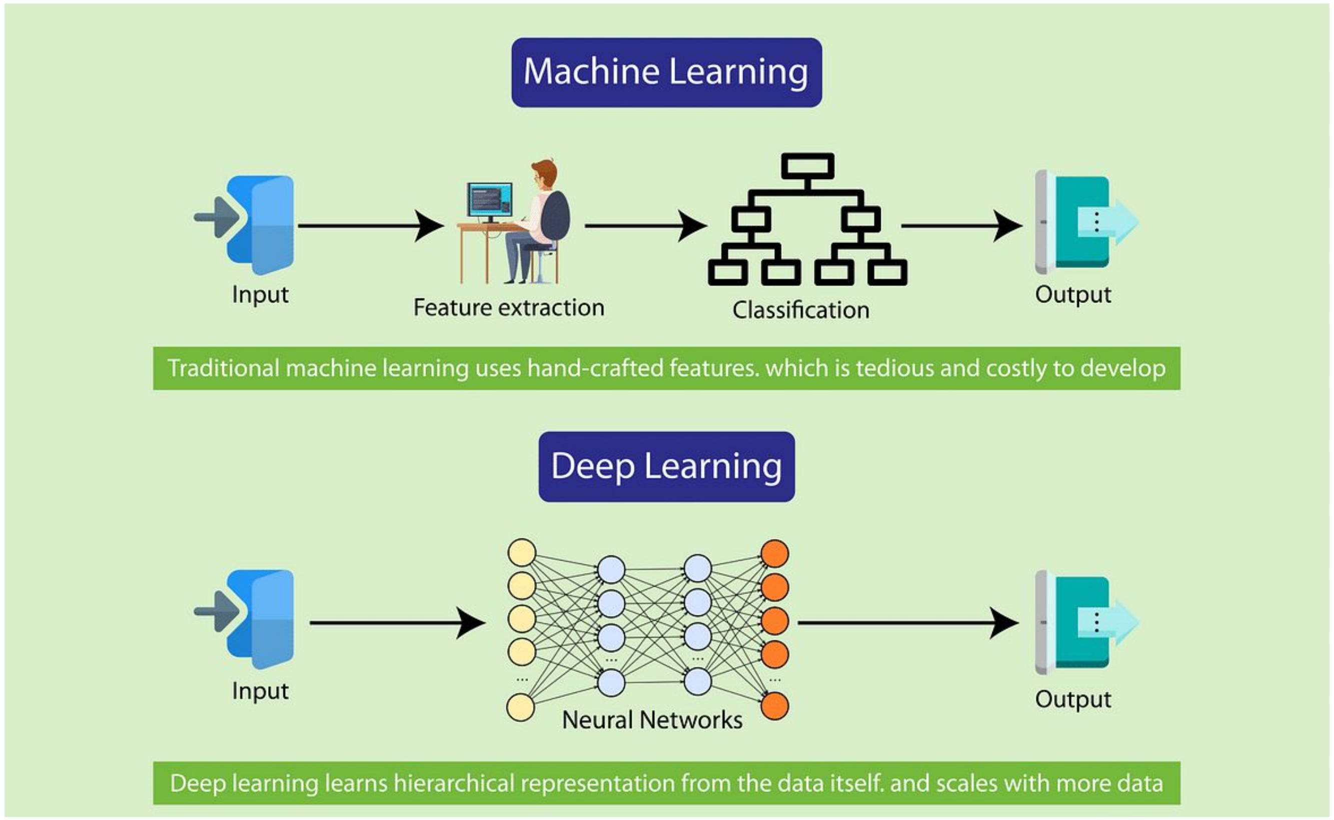
- Machine Learning comes up with the decision making "program" (machine learning model) that optimizes the decision making according to a user-defined objective.

Artificial Intelligence Overview


ARTIFICIAL INTELLIGENCE
Early artificial intelligence stirs excitement.
MACHINE LEARNING
Machine learning begins to flourish.
DEEP LEARNING
Deep learning breakthroughs drive AI boom.
1950's
1960's
1970's
1980's
1990's
2000's
2010's

Since an early flush of optimism in the 1950s, smaller subsets of artificial intelligence – first machine learning, then deep learning, a subset of machine learning – have created ever larger disruptions.

Machine Learning vs Deep Learning

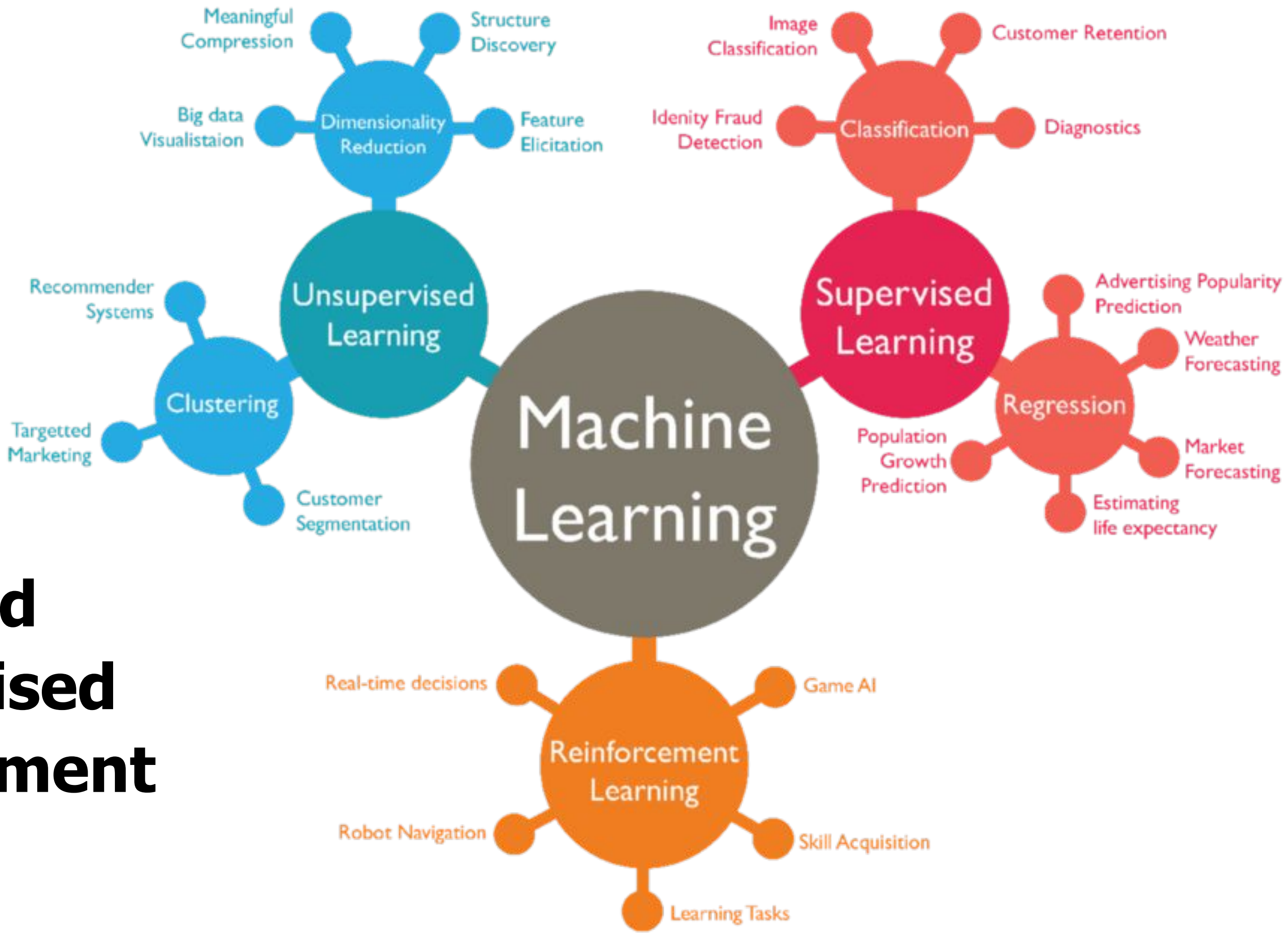

Machine Learning
Input
Feature extraction
Classification
Output
Traditional machine learning uses hand-crafted features. which is tedious and costly to develop
Deep Learning
Input
Neural Networks
Output
Deep learning learns hierarchical representation from the data itself. and scales with more data

- Deep learning models are gaining popularity for the fact that they can achieve state-of-the-art accuracy which at times can **outperform human**

What is Machine Learning (ML)?

- ML allows machines(computers) to learn from data or experience and make a prediction based on the experience
- ML enables the computers or the machines to make data-driven decisions rather than being explicitly programmed for carrying out a certain task
- These programs or algorithms are designed in a way that they learn and improve over time when are exposed to new data

Types of Machine Learning



- **Supervised**
- **Unsupervised**
- **Reinforcement**

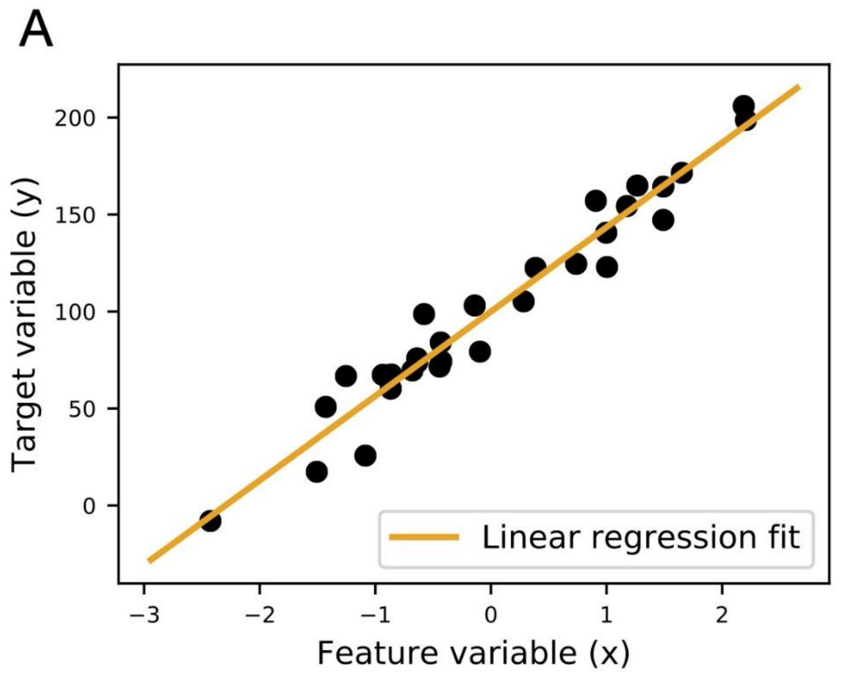
Supervised Learning

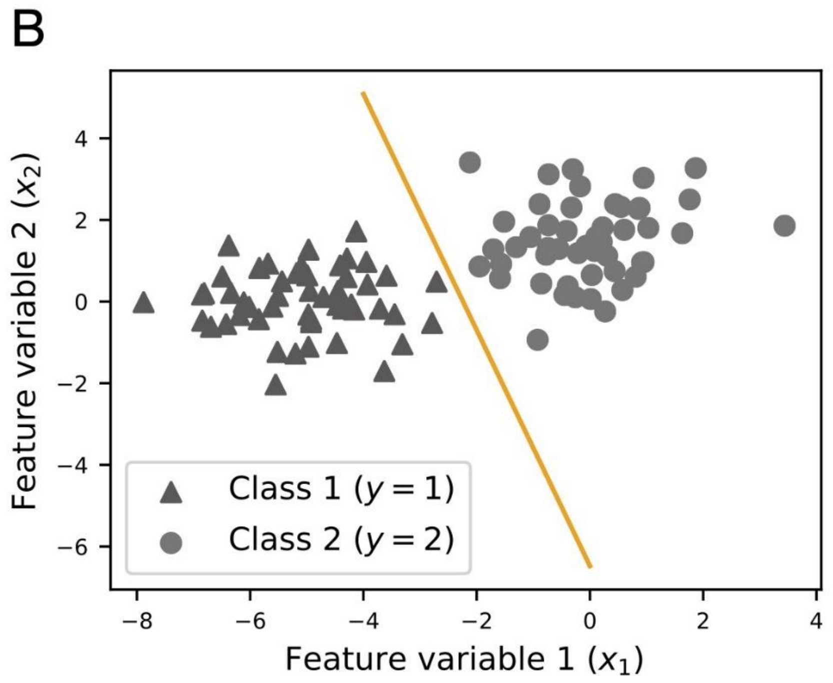
- Find the mapping between the input variable(X) and output variable(Y) (input X and label Y)

- Supervised learning is divided into two types:
  - Regression — The output variable is continuous and real value. For example price, weight, etc

  - Classification — The output variable is a category, such as "red" or "blue" or "disease" or "no disease".

Regression and Classification



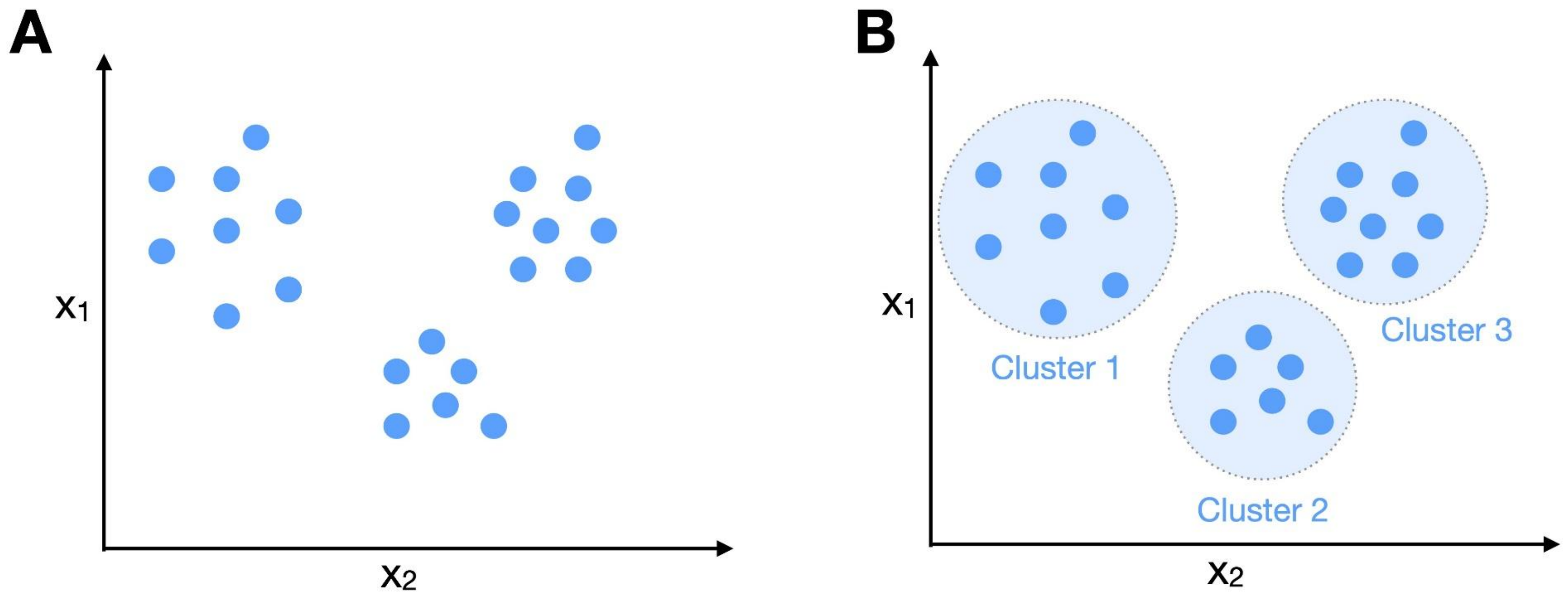


llustrations of the two main categories of supervised learning, regression (A) and classification (B)

Unsupervised Learning

- Unsupervised learning is where you only have input data (X) and no corresponding output variables. The goal of unsupervised learning is to model the **underlying structure or distribution in the data** in order to learn more about the data.
- Unsupervised learning is divided into two types:
  - Clustering: discover the inherent groupings in the data, such as grouping customers by purchasing behavior.
  - Association: discover rules that describe large portions of your data, such as people that buy X also tend to buy Y.

Example of Clustering



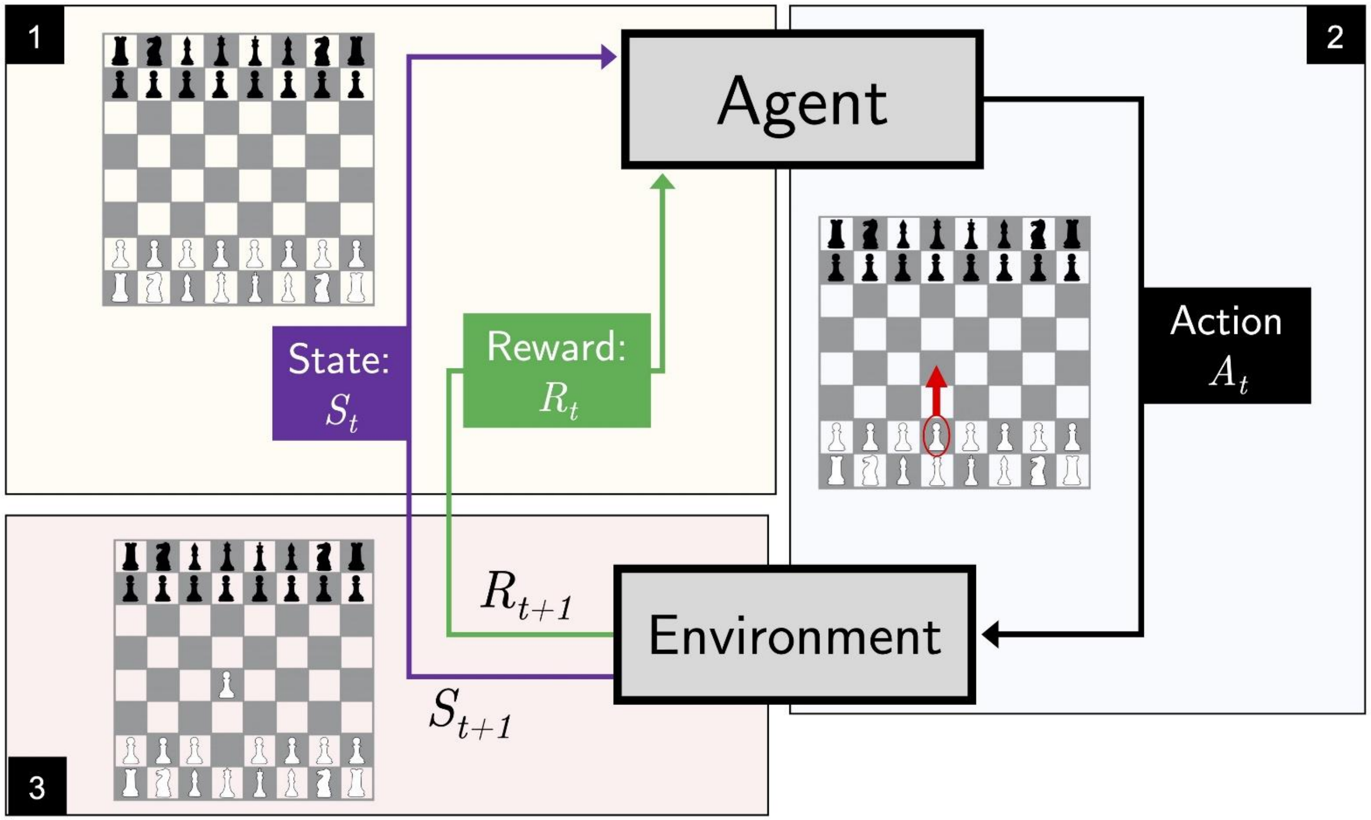
- It seems that there are 3 clusters for the example dataset

Reinforcement Learning

- Reinforcement learning, in the context of artificial intelligence, is a type of dynamic programming that trains algorithms using a system of reward and punishment

- The agent receives rewards by performing correctly and penalties for performing incorrectly. The agent learns without intervention from a human by maximizing its reward and minimizing its penalty

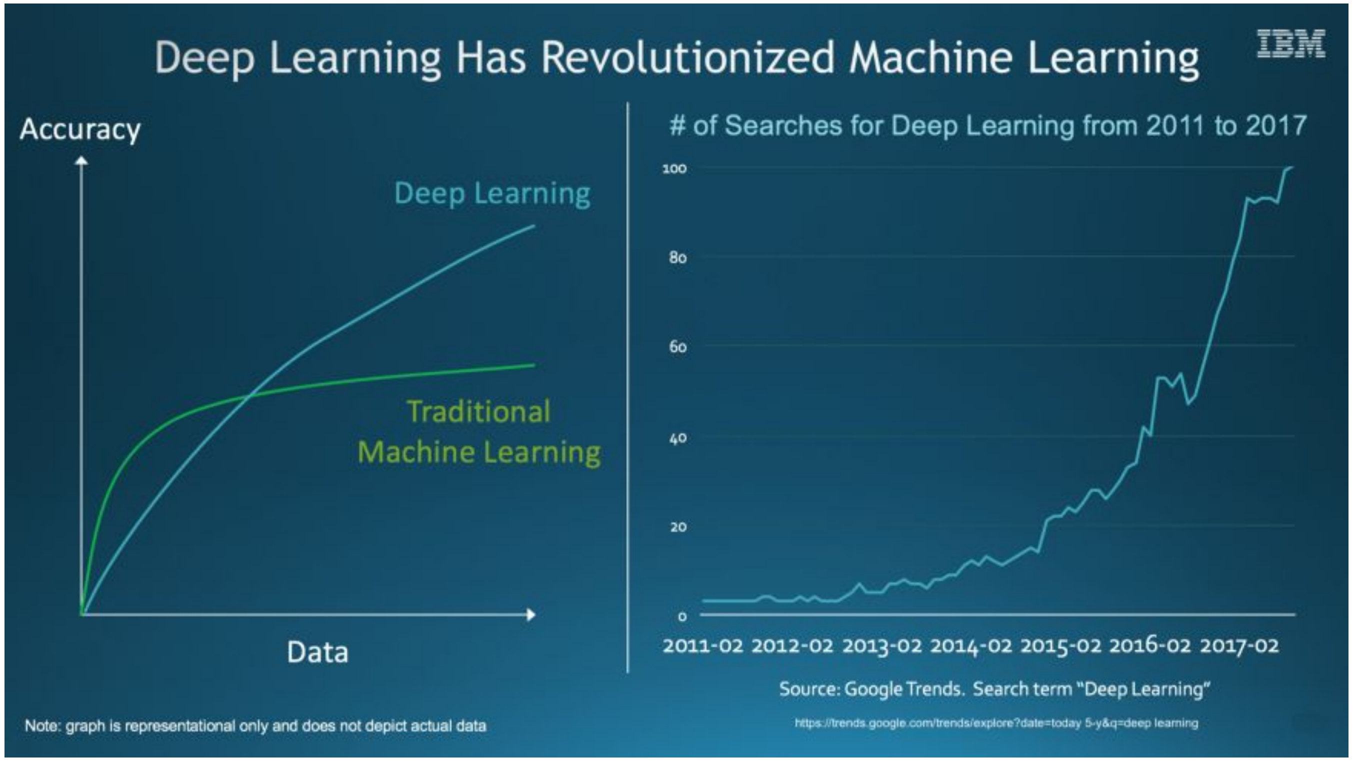
Example of Reinforcement



- AI Games (Bot)
- Optimize the power consumption in a house

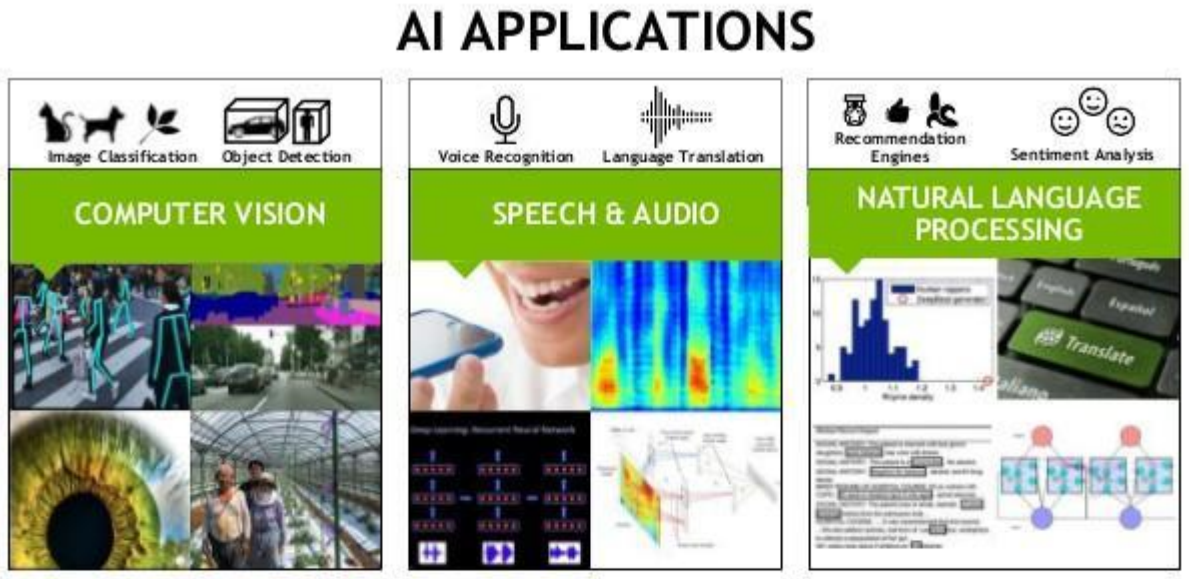
What is Deep Learning?

- Deep Learning is a subfield of machine learning concerned with algorithms inspired by the structure and function of the brain called artificial neural networks.



Applications of Deep Learning

- Self-driving cars
- Voice search and virtual assistants
- Machine translation
- Image caption generation
- Real-time object recognition in the image (Google lens).

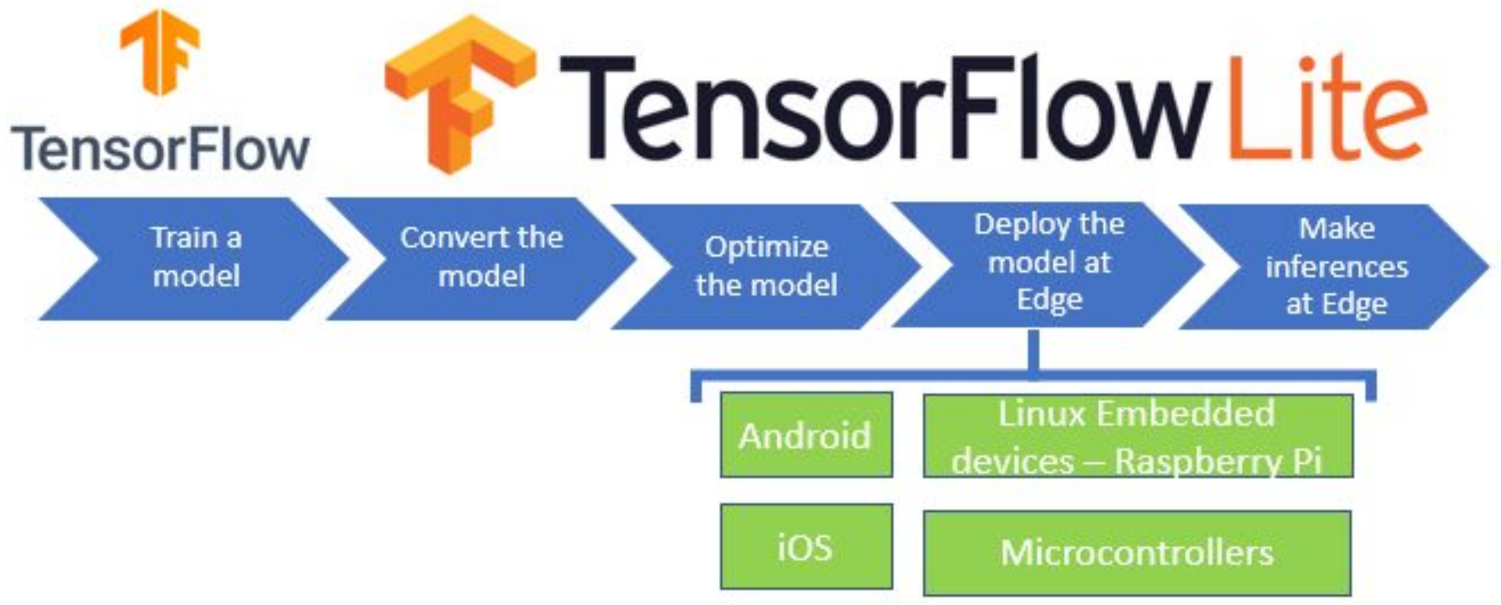


Introduction to TensorFlow

What is TensorFlow?

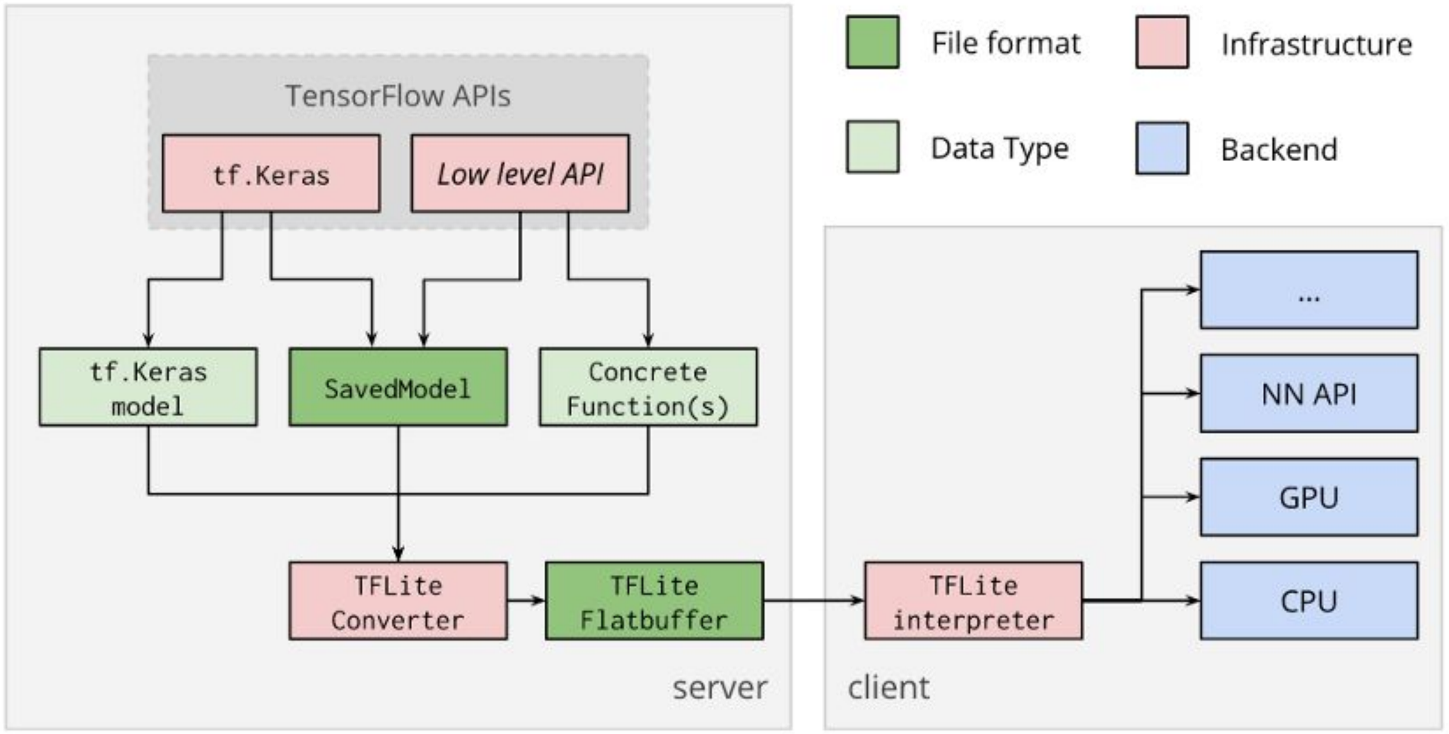
- TensorFlow is one of the most popular machine learning frameworks that help engineers, deep neural scientists to create AI models (Machine Learning and Deep Learning)

TensorFlow Lite



- TensorFlow Lite is an open-source, product ready, cross-platform deep learning framework that converts a pre-trained model in TensorFlow to a special format that can be optimized for speed or storage.

Convert TensorFlow and TensorFlow Lite



- Edge Computing and AI

Build A Simple AI using Google Teachable Machine and Python

LÊ TRỌNG NHÂN

trongnhanle@hcmut.edu.vn
trongnhanle85@gmail.com

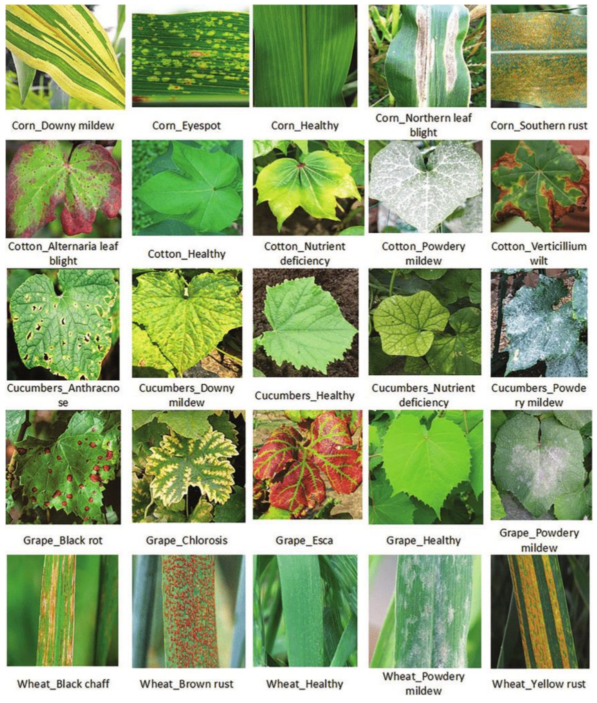
Content

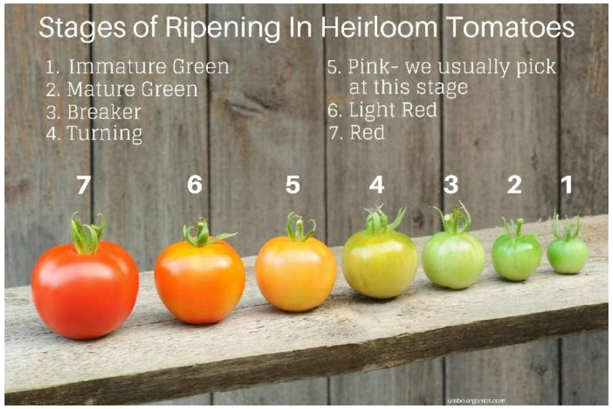
- Build a simple AI model in Google Teachable Machine
- Implement AI inference using Python
- Extra work: Publish AI result to Adafruit IO server

Part 1: Build A Simple AI

Build a simple AI model

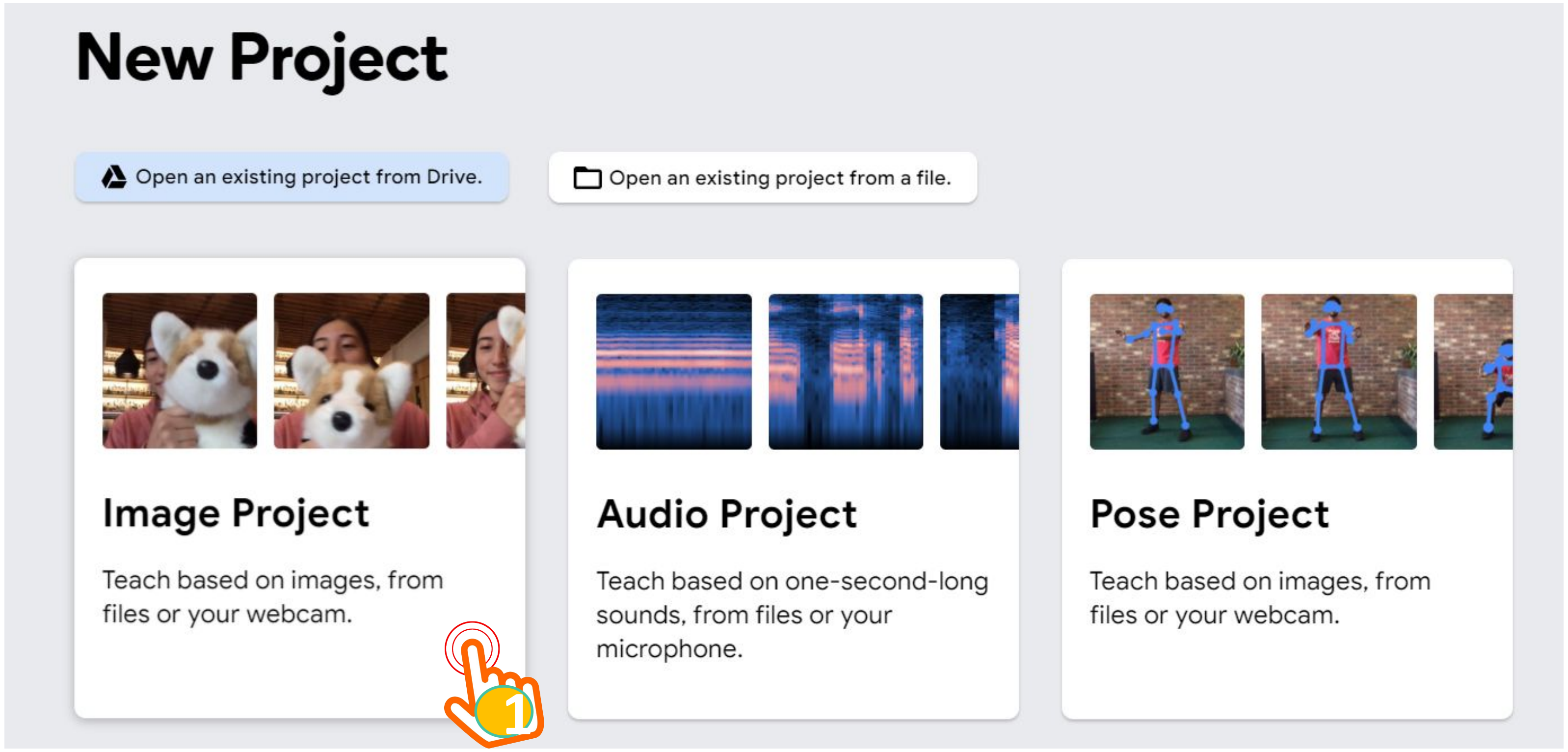
- Use PC webcam to recognize simple objects, for instance:
  - People is wearing masks or not wearing masks
  - Ripe tomatoes (normally in red) or raw tomatoes (normally in green)
  - Leaves are wilted or diseased




Stages of Ripening In Heirloom Tomatoes
1. Immature Green
2. Mature Green
3. Breaker
4. Turning
5. Pink- we usually pick at this stage
6. Light Red
7. Red
7
6
5
4
3
2
1

Step 1

- Go to the Google Teachable Machine website:
  - https://teachablemachine.withgoogle.com/train



New Project

Open an existing project from Drive.
Open an existing project from a file.
1

Image Project

Teach based on images, from files or your webcam.

Audio Project

Teach based on one-second-long sounds, from files or your microphone.

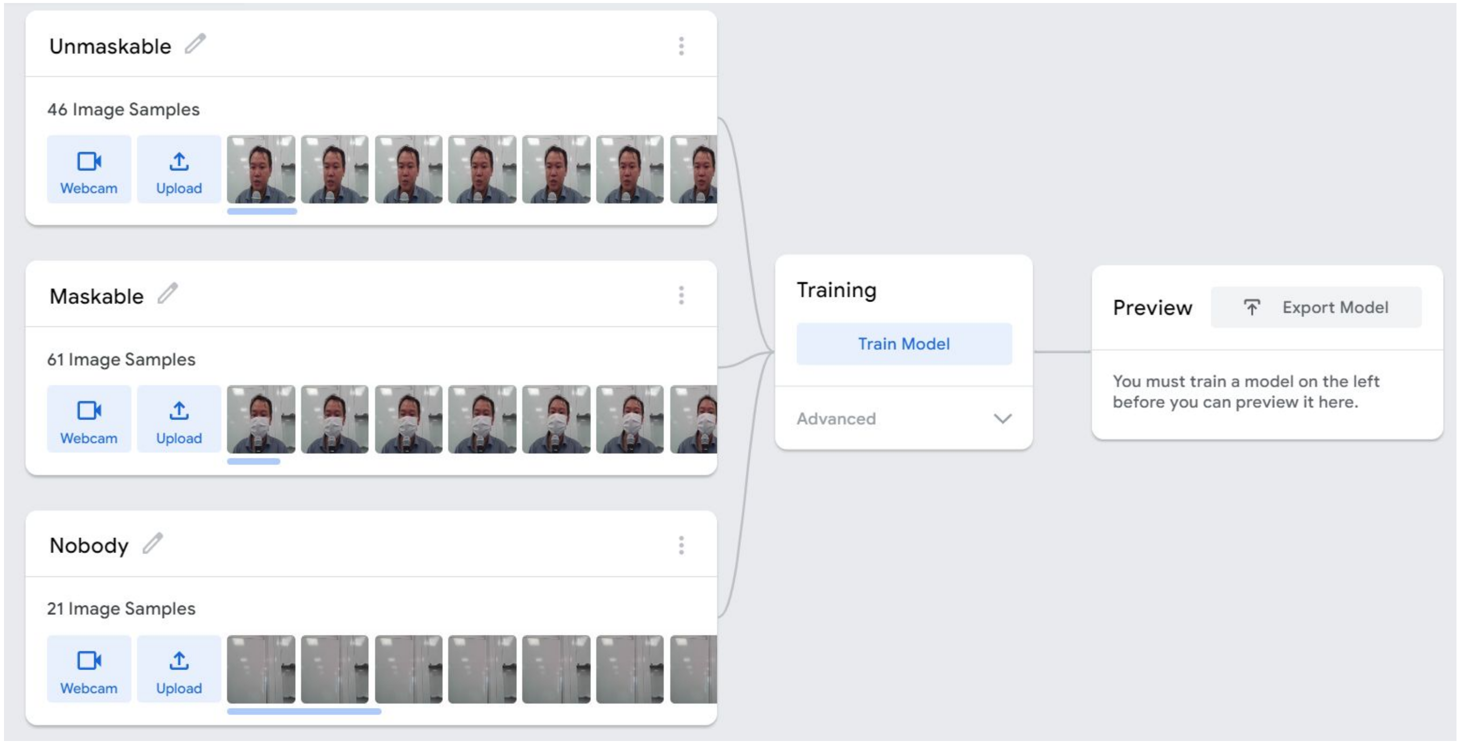
Pose Project

Teach based on images, from files or your webcam.

- Select on the **Image Project**

Step 2

- Provide images for each class. Note that each class should have a name.

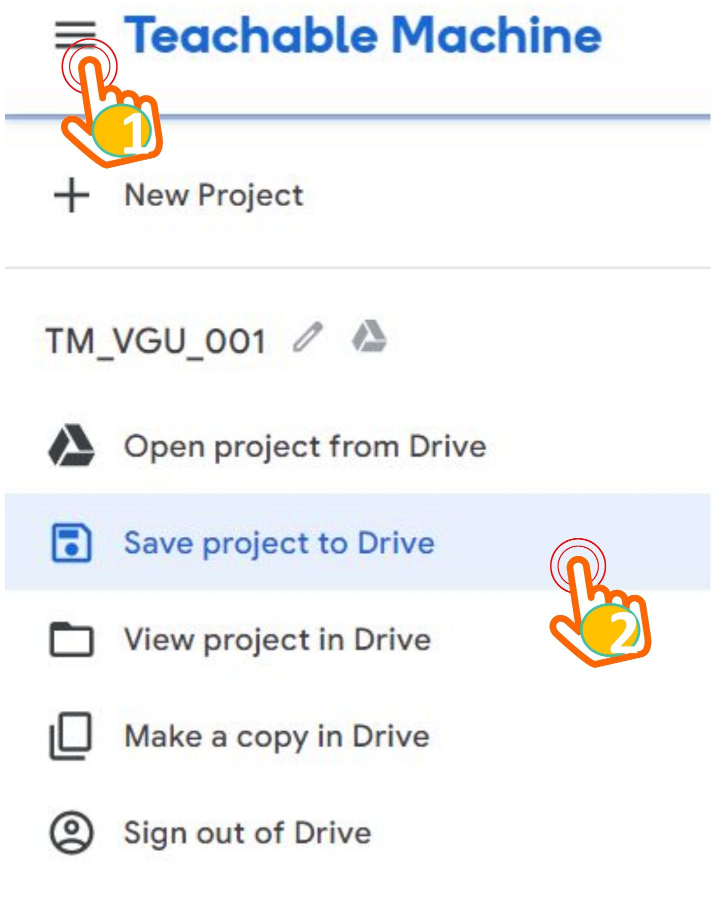


You must train a model on the left before you can preview it here.

- Provide a class name “**background**”, when there is no object need to be recognized

Step 3

- Save your project to Google Drive



Teachable Machine

1

New Project

TM_VGU_001

Open project from Drive

Save project to Drive

2

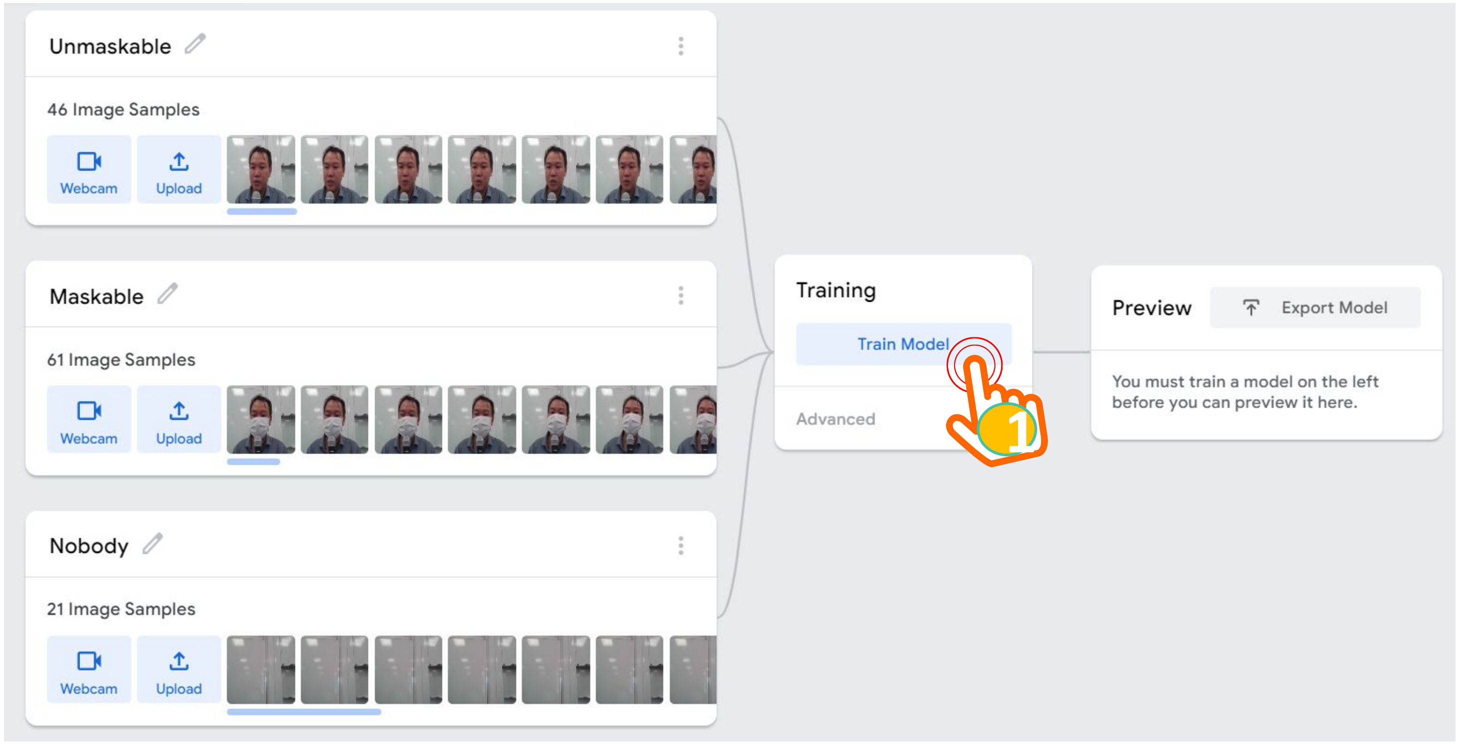
View project in Drive

Make a copy in Drive

Sign out of Drive

Step 4

- Click on the Train Model button to train your AI

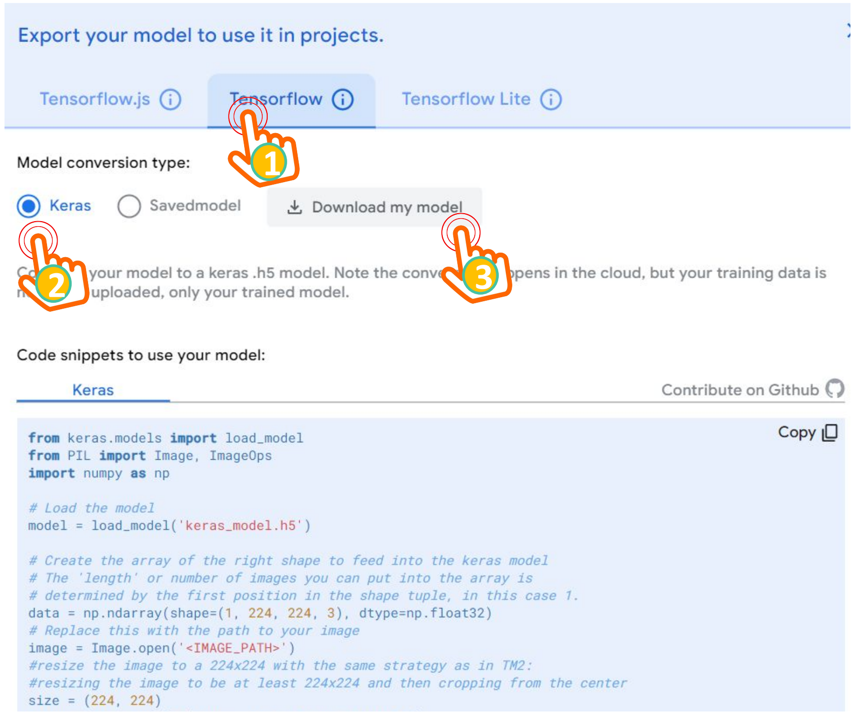


You must train a model on the left before you can preview it here.

- Test your model in the Preview option. If the performance is not so good, provide more images and train the AI again

Step 4

- Export the AI model:
  - Click on Export
  - Select Tensorflow, Keras and Download the Model



Co your model to a keras .h5 model. Note the conve pens in the cloud, but your training data is n uploaded, only your trained model.

```
from keras.models import load_model
from PIL import Image, ImageOps
import numpy as np

# Load the model
model = load_model('keras_model.h5')

# Create the array of the right shape to feed into the keras model
# The 'length' or number of images you can put into the array is
# determined by the first position in the shape tuple, in this case 1.
data = np.ndarray(shape=(1, 224, 224, 3), dtype=np.float32)
# Replace this with the path to your image
image = Image.open('<IMAGE_PATH>')
#resize the image to a 224x224 with the same strategy as in TM2:
#resizing the image to be at least 224x224 and then cropping from the center
size = (224, 224)
```

Step 5

- Prepare materials for Python programming:
  - Unzip the files
  - Copy the source code from Google Teachable to Notepad

Python Source Code

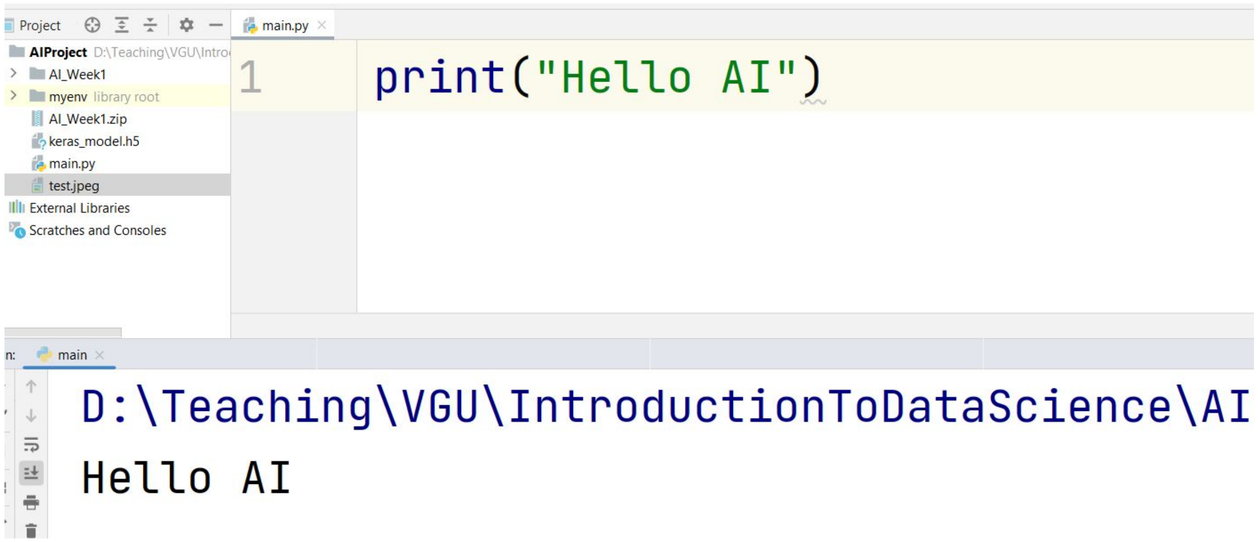
```
from keras.models import load_model
from PIL import Image, ImageOps
import numpy as np

# Load the model
model = load_model('keras_model.h5')

# Create the array of the right shape to feed into the keras model
# The 'length' or number of images you can put into the array is
# determined by the first position in the shape tuple, in this case 1.
data = np.ndarray(shape=(1, 224, 224, 3), dtype=np.float32)
# Replace this with the path to your image
image = Image.open('<IMAGE_PATH>')
#resize the image to a 224x224 with the same strategy as in TM2:
#resizing the image to be at least 224x224 and then cropping from the center
size = (224, 224)
image = ImageOps.fit(image, size, Image.ANTIALIAS)

#turn the image into a numpy array
image_array = np.asarray(image)
# Normalize the image
normalized_image_array = (image_array.astype(np.float32) / 127.0) - 1
# Load the image into the array
data[0] = normalized_image_array

# run the inference
prediction = model.predict(data)
print(prediction)
```

AI Files (keras_model.h5 and lables.txt)

Part 2: Python Programming

Step 1

- Create a new Python project:
  - Save the project in an empty folder (e.g. AI_Python)
  - Python version 3.8.5 (or older) is recommended



```
print("Hello AI")
```

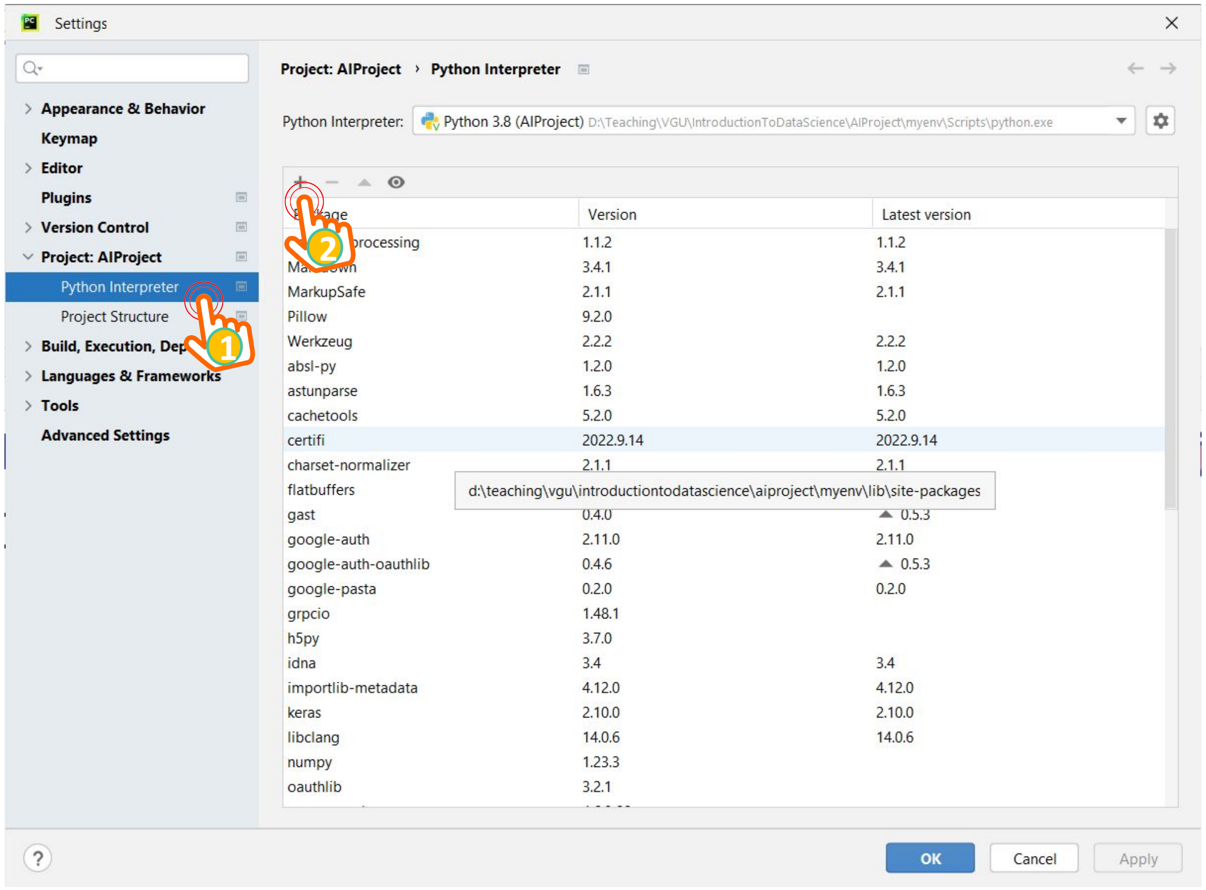
```
D:\Teaching\VGU\IntroductionToDataScience\AI
Hello AI
```

  - Student should print to the console a simple string in order to test the Python Editor and Python Interpreter

Step 2

- Install following libraries (required for AI processing):
  - tensorflow
  - keras
  - numpy
  - pillow
  - opencv-python
- In Pycharm Editor, these libraries can be installed through GUI:
  - From menu File, select Setting
  - Navigate to the Project
  - Select Python Interpreter
  - Click on Add icon, filter the library and click on Install

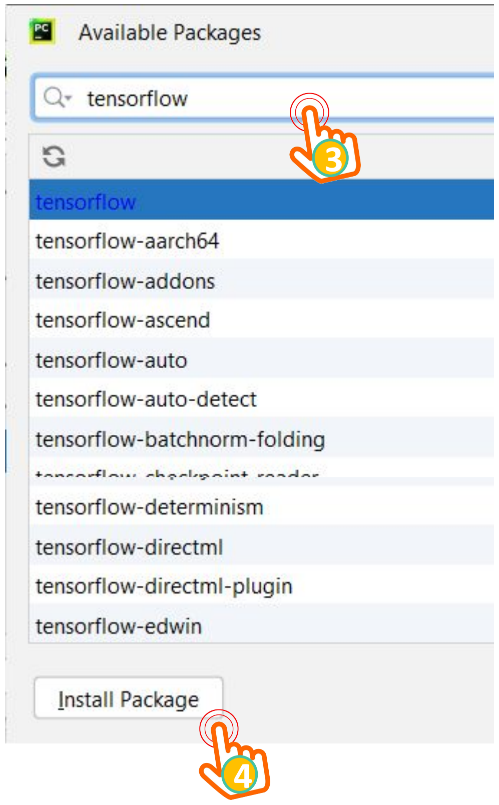
Install library by GUI in PyCharm



Python Interpreter: Python 3.8 (AIProject) D:\Teaching\VGU\IntroductionToDataScience\AIProject\myenv\Scripts\python.exe

Package	Version	Latest version
processing	1.1.2	1.1.2
Ma own	3.4.1	3.4.1
MarkupSafe	2.1.1	2.1.1
Pillow	9.2.0	
Werkzeug	2.2.2	2.2.2
absl-py	1.2.0	1.2.0
astunparse	1.6.3	1.6.3
cachetools	5.2.0	5.2.0
certifi	2022.9.14	2022.9.14
charset-normalizer	2.1.1	2.1.1
flatbuffers		
gast	0.4.0	▲ 0.5.3
google-auth	2.11.0	2.11.0
google-auth-oauthlib	0.4.6	▲ 0.5.3
google-pasta	0.2.0	0.2.0
grpcio	1.48.1	
h5py	3.7.0	
idna	3.4	3.4
importlib-metadata	4.12.0	4.12.0
keras	2.10.0	2.10.0
libclang	14.0.6	14.0.6
numpy	1.23.3	
oauthlib	3.2.1	

d:\teaching\vgu\introductiontodatascience\aiproject\myenv\lib\site-packages



tensorflow

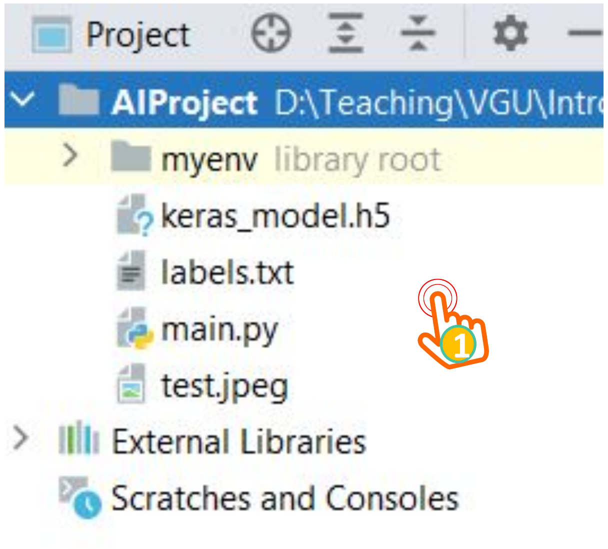
- tensorflow
- tensorflow-aarch64
- tensorflow-addons
- tensorflow-ascend
- tensorflow-auto
- tensorflow-auto-detect
- tensorflow-batchnorm-folding
- tensorflow-checkpoint-reader
- tensorflow-determinism
- tensorflow-directml
- tensorflow-directml-plugin
- tensorflow-edwin

- This option is only available in Pycharm IDE. Other IDEs normally use the pip command to install a package

Step 4

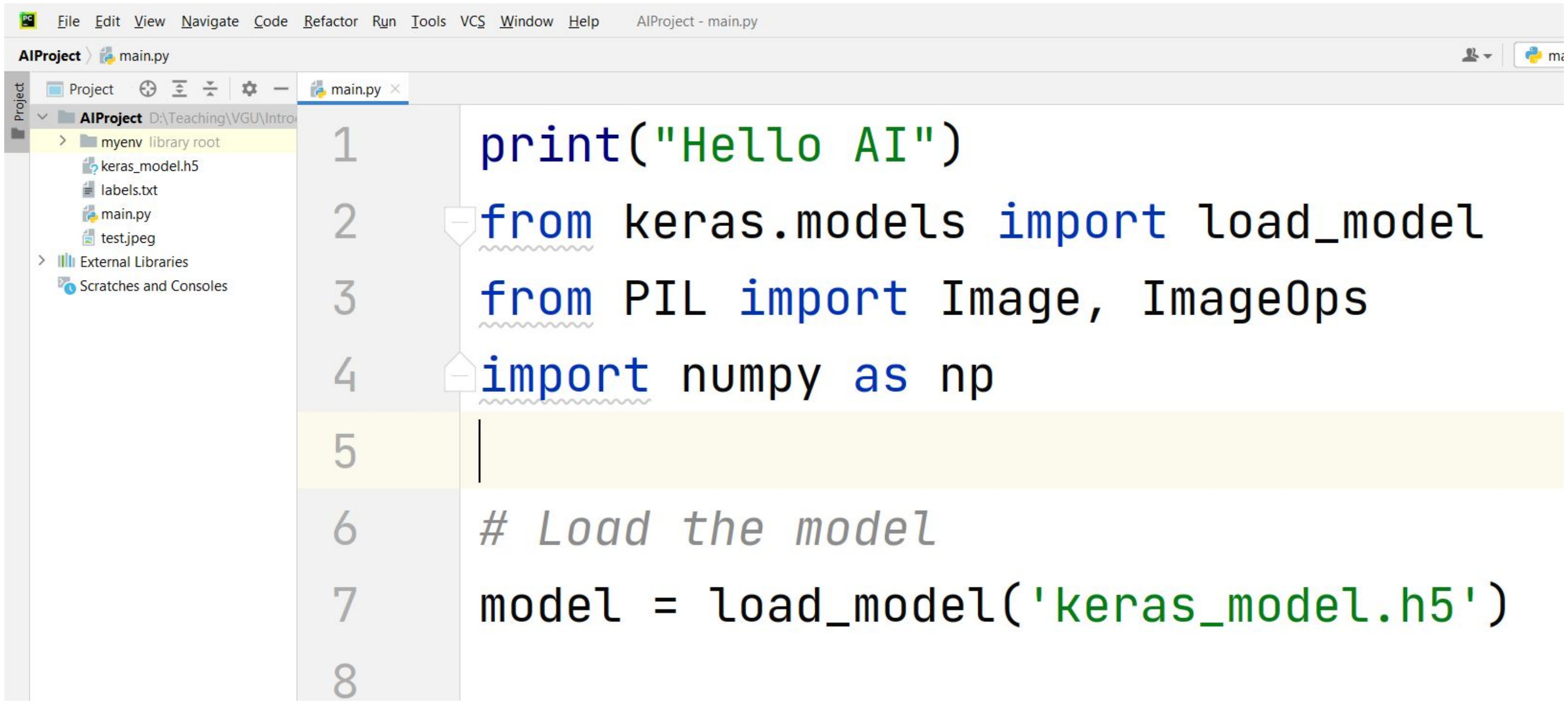
- Copy the AI Files (keras_model.h5 and lables.txt) to the same folder of main.py

- Copy an simple image to the same folder of main.py (the resolution should be higher than 224x224 pixel)



Step 5

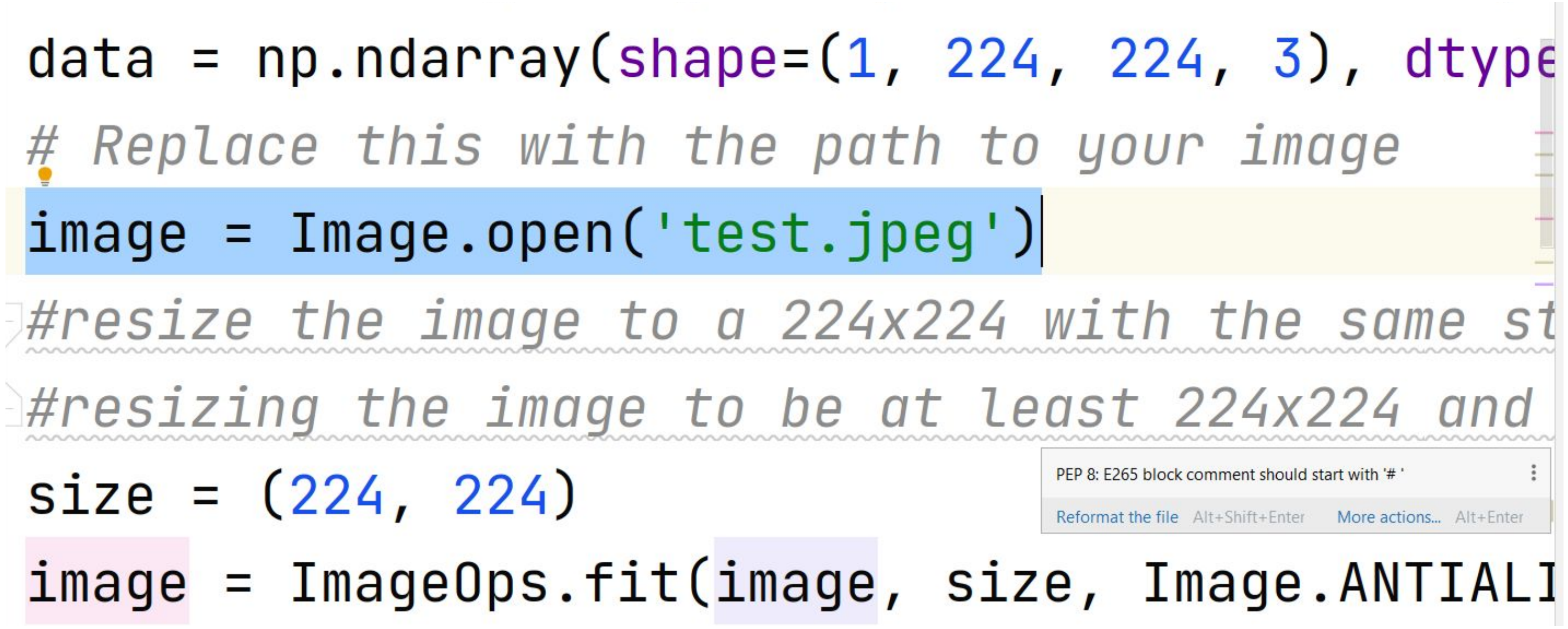
- Copy the whole source code and place in main.py:



```python
print("Hello AI")
from keras.models import load_model
from PIL import Image, ImageOps
import numpy as np

# Load the model
model = load_model('keras_model.h5')

```

Step 6

- Modify the line 13 or 14, change the name of the image to your image, which is placed in the same folder with main.py

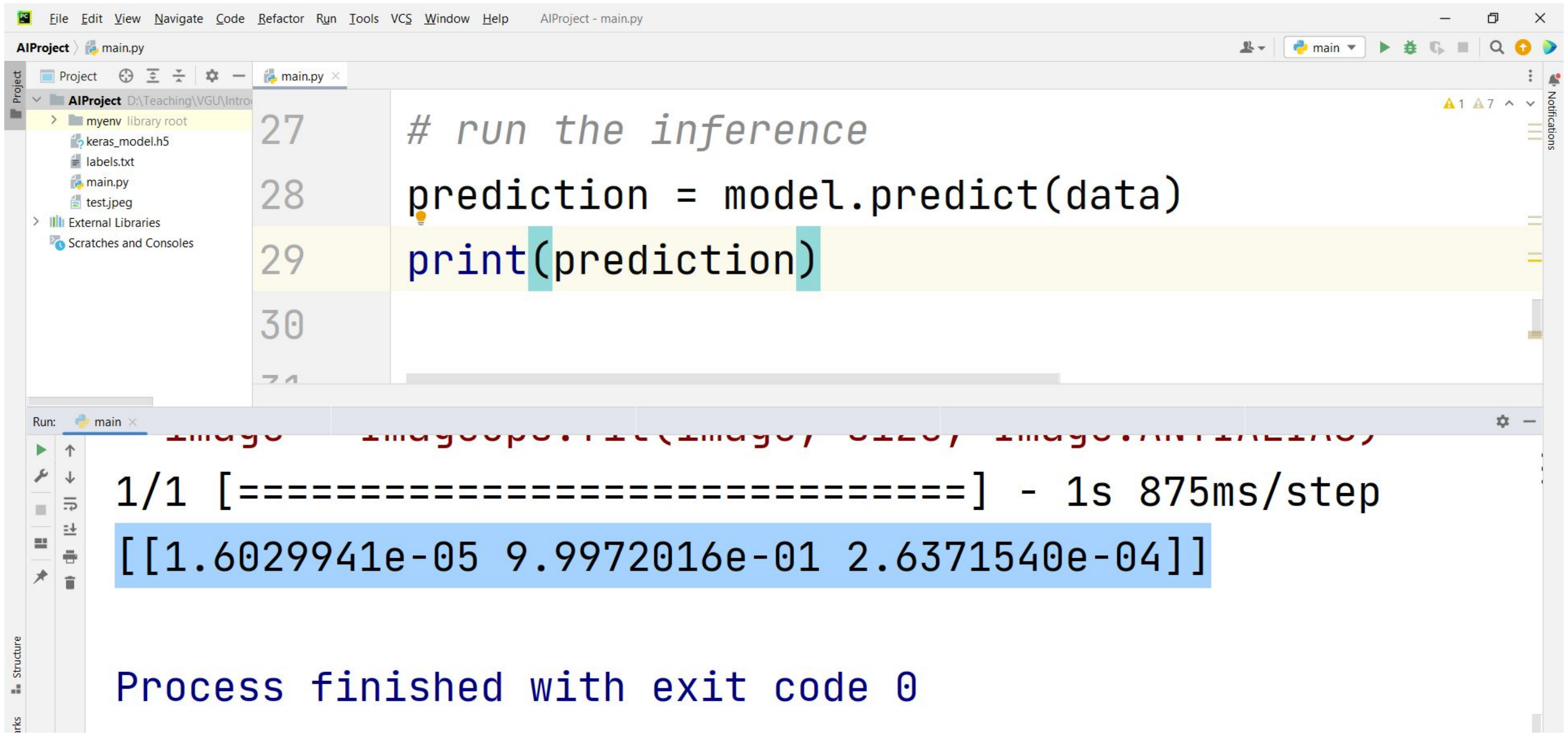


```
data = np.ndarray(shape=(1, 224, 224, 3), dtype
# Replace this with the path to your image
image = Image.open('test.jpeg')
#resize the image to a 224x224 with the same st
#resizing the image to be at least 224x224 and
size = (224, 224)
image = ImageOps.fit(image, size, Image.ANTIALI
```

PEP 8: E265 block comment should start with '# '

Step 7

- Run the python program. The output should be an 2D array, presenting the **confidence (in percent)** of all the labels trained in Google Teachable Machine



```
# run the inference
prediction = model.predict(data)
print(prediction)
```

```
1/1 [==============================] - 1s 875ms/step
[[1.6029941e-05 9.9972016e-01 2.6371540e-04]]

Process finished with exit code 0
```

Step 8

- Find the maximum confidence and its index from 2D array

```
#get the 1D array
output = prediction[0]
#assign default value for max confidence
max_index = 0
max_confidence = output[0]
#find the maximum confidence and its index
for i in range(1, len(output)):
    if max_confidence < output[i]:
        max_confidence = output[i]
        max_index = i
print(max_index, max_confidence)
```

Step 9

- Map the labels to the maximum index

```
file = open("labels.txt",encoding="utf8")
data = file.read().split("\n")
print("AI Result: ", data[max_index])
```

Step 10

- Add live image from the camera

```
import cv2
cam = cv2.VideoCapture(0)

def image_capture():
    ret,frame =
cam.read()
    cv2.imwrite
("test.png",frame)
```

Step 11

- Refactor the python source code, having image_capture() and image_detector()

```
print("Hello AI")
from keras.models import load_model
from PIL import Image, ImageOps
import numpy as np

import cv2
cam = cv2.VideoCapture(0)
# Load the model
model = load_model('keras_model.h5')

def image_capture():
   ret,frame = cam.read()
   cv2.imwrite ("test.png",frame)

def image_detector():
    data = np.ndarray(shape=(1, 224, 224, 3), dtype=np.float32)
    # Replace this with the path to your image
    image = Image.open('test.png')
    #resize the image to a 224×224 with the same strategy as in TM2:
    #resizing the image to be at least 224×224 and then cropping
from the center
    size = (224, 224)
    image = ImageOps.fit(image, size, Image.ANTIALIAS)

    #turn the image into a numpy array
    image_array = np.asarray(image)
    # Normalize the image
    normalized_image_array = (image_array.astype(np.float32) /
127.0) - 1
    # Load the image into the array
    data[0] = normalized_image_array

    # run the inference
    prediction = model.predict(data)

    #get the 1D array
    output = prediction[0]
    #assign default value for max confidence
    max_index = 0
    max_confidence = output[0]
    #find the maximum confidence and its index
    for i in range(1, len(output)):
        if max_confidence < output[i]:
            max_confidence = output[i]
            max_index = i
    print(max_index, max_confidence)

    file = open("labels.txt",encoding="utf8")
    data = file.read().split("\n")
    print("AI Result: ", data[max_index])
```

Step 12

- Add an infinite loop to the python source code.
- Import time library
- Every 5 seconds, call image_capture() then call image_detector()

```python
print("Hello AI")
from keras.models import load_model
from PIL import Image, ImageOps
import numpy as np
import cv2
import time

cam = cv2.VideoCapture(0)
# Load the model
model = load_model('keras_model.h5')

def image_capture():
   ret,frame = cam.read()
   cv2.imwrite ("test.png",frame)

def image_detector():

    data = np.ndarray(shape=(1, 224, 224, 3),
dtype=np.float32)
    # Replace this with the path to your image
    image = Image.open('test.png')
    #resize the image to a 224x224 with the same strategy
as in TM2:
    #resizing the image to be at least 224x224 and then
cropping from the center
    size = (224, 224)
    image = ImageOps.fit(image, size, Image.ANTIALIAS)

    #turn the image into a numpy array
    image_array = np.asarray(image)
    # Normalize the image
    normalized_image_array =
(image_array.astype(np.float32) / 127.0) - 1
    # Load the image into the array
    data[0] = normalized_image_array

    # run the inference
    prediction = model.predict(data)

    #get the 1D array
    output = prediction[0]
    #assign default value for max confidence
    max_index = 0
    max_confidence = output[0]
    #find the maximum confidence and its index
    for i in range(1, len(output)):
        if max_confidence < output[i]:
            max_confidence = output[i]
            max_index = i
    print(max_index, max_confidence)

    file = open("labels.txt",encoding="utf8")
    data = file.read().split("\n")
    print("AI Result: ", data[max_index])

while True:
      time.sleep(5)
      image_capture()
```

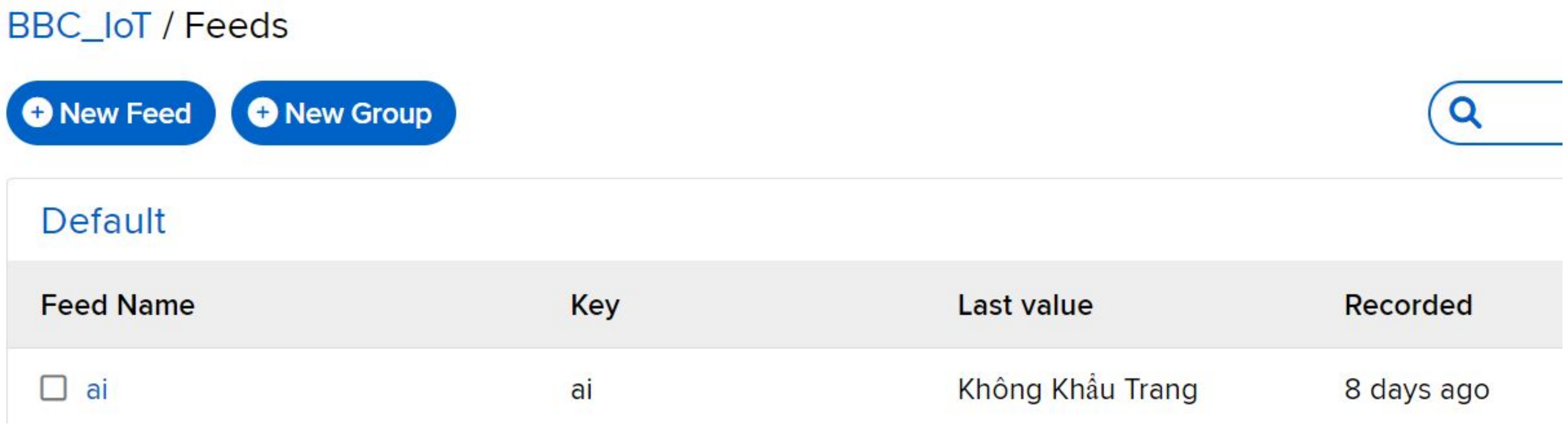
Part 3: Upload to Adafruit IO

Introduction

- The AI result can be published to the Adafruit IO for remote monitoring
- When AI is combined with IoT, we form a kind of AIoT application!!!

Step 1

- Create an account in Adafruit IO
- Create a feed on Adafruit IO:
  - Feed/ New Feed
  - Provide the name for the Feed, e.g. "ai"

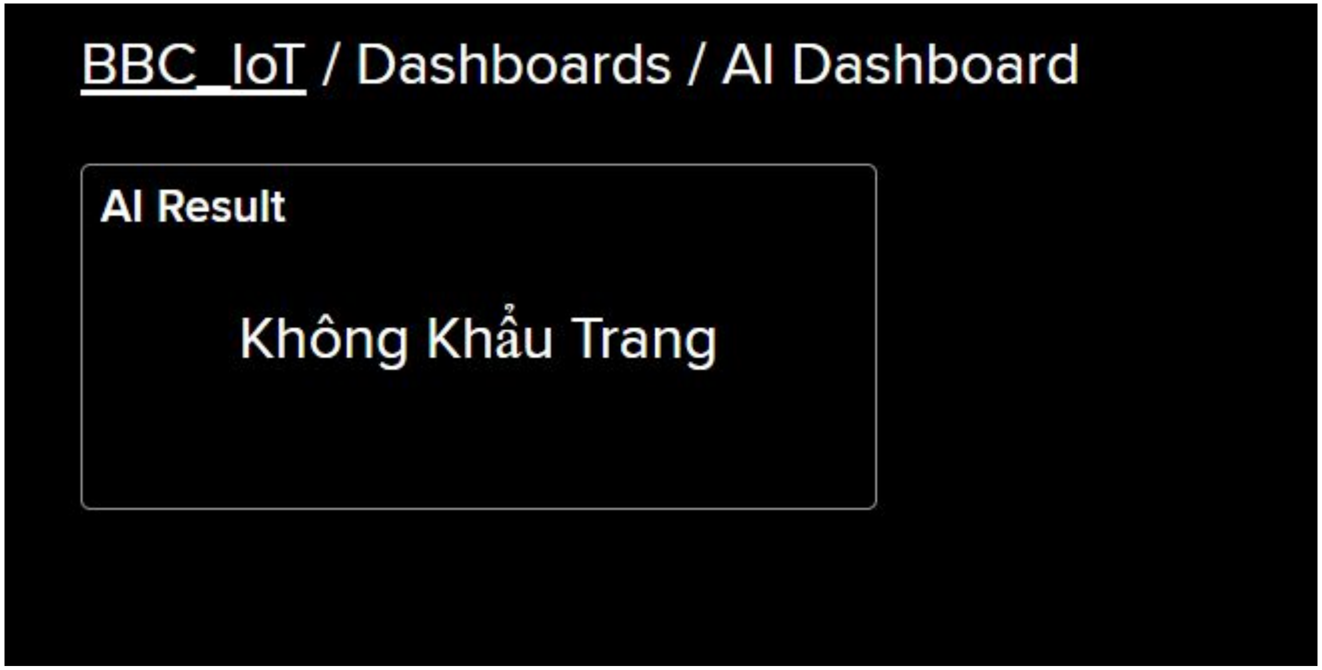

BBC_IoT / Feeds

Default

Feed Name	Key	Last value	Recorded
ai	ai	Không Khẩu Trang	8 days ago

Step 2

- Create a simple Dashboard, have a textview to display the last data from feed ai

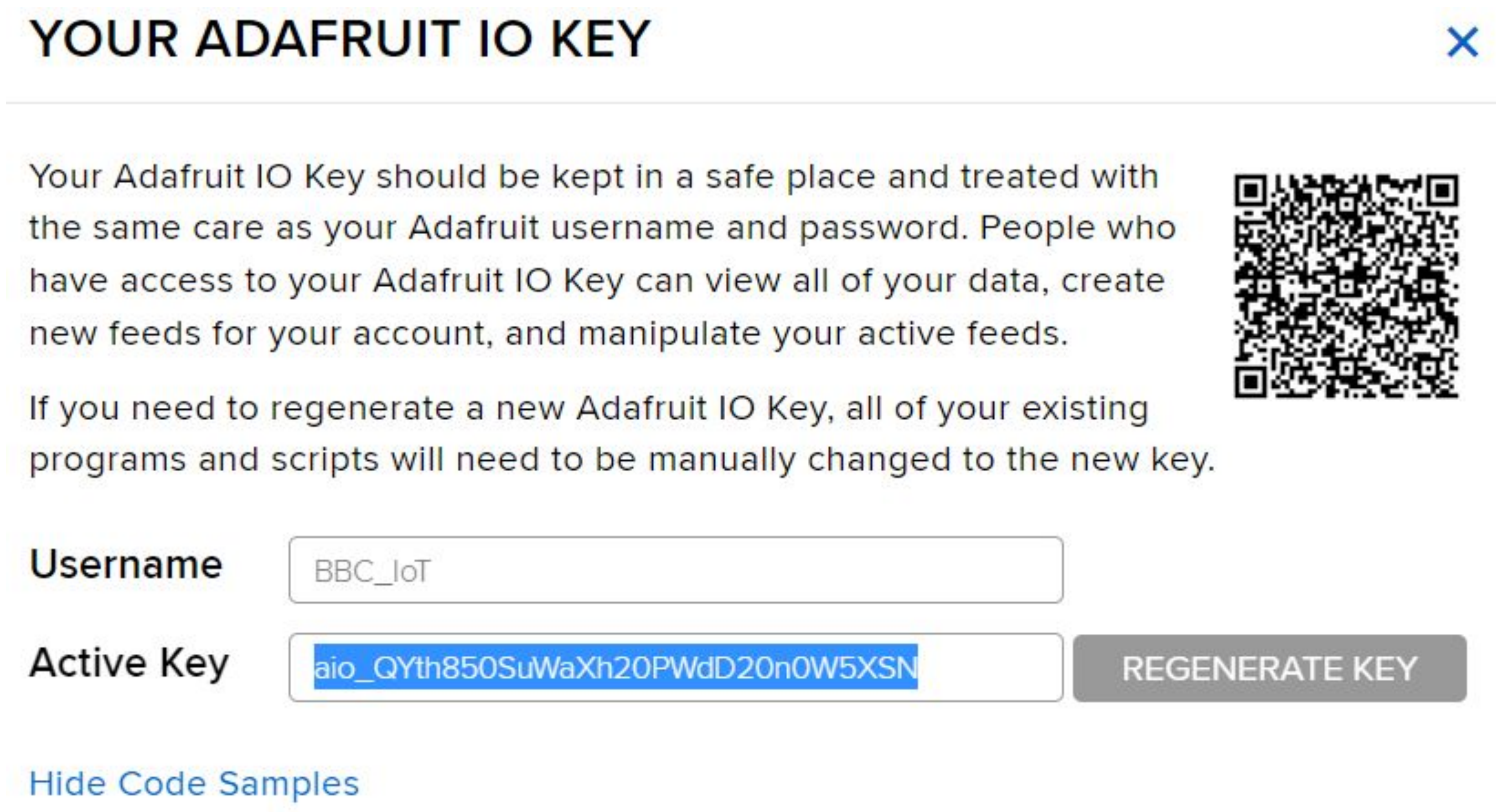

BBC_IoT / Dashboards / AI Dashboard

AI Result

Không Khẩu Trang

Step 3

- Copy your account and your key, which are used for the Python programming



YOUR ADAFRUIT IO KEY

Your Adafruit IO Key should be kept in a safe place and treated with the same care as your Adafruit username and password. People who have access to your Adafruit IO Key can view all of your data, create new feeds for your account, and manipulate your active feeds.

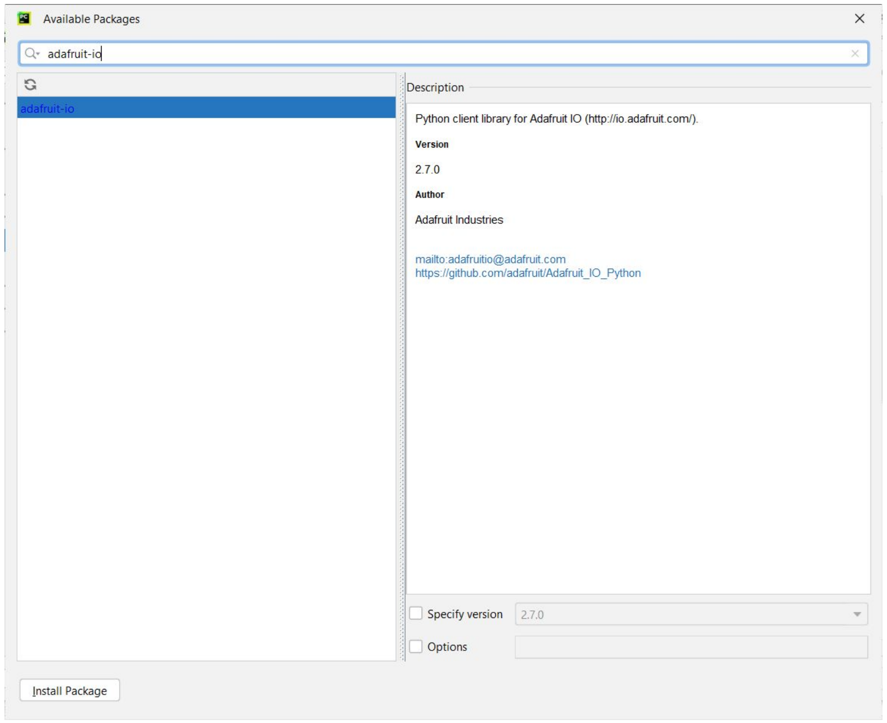
If you need to regenerate a new Adafruit IO Key, all of your existing programs and scripts will need to be manually changed to the new key.

Username	BBC_IoT
Active Key	aio_QYth850SuWaXh20PWdD20n0W5XSN

Step 4

- Install the **adafruit-io** library
- Import libs:

```
import sys
from Adafruit_IO import MQTTClient
```



Python client library for Adafruit IO (http://io.adafruit.com/).

Version

2.7.0

Author

Adafruit Industries

mailto:adafruitio@adafruit.com
https://github.com/adafruit/Adafruit_IO_Python

Step 5

- Create an instance of the MQTT Client

```
client = MQTTClient("your_usr","your_key")

client.connect()
client.loop_background()
```

- Then, use client.publish(“ai”, “your_ai_result”) to the service

Step 6

- Finalize the whole program

```
print("Hello AI")
from keras.models import load_model
from PIL import Image, ImageOps
import numpy as np
import cv2
import time
from Adafruit_IO import MQTTClient

cam = cv2.VideoCapture(0)
# Load the model
model = load_model('keras_model.h5')

def image_capture():
   ret,frame = cam.read()
   cv2.imwrite ("test.png",frame)

def image_detector():
    # Create the array of the right shape to feed into the
keras model
    # The 'length' or number of images you can put into the
array is
    # determined by the first position in the shape tuple,
in this case 1.
    data = np.ndarray(shape=(1, 224, 224, 3),
dtype=np.float32)
    # Replace this with the path to your image
    image = Image.open('test.png')
    #resize the image to a 224x224 with the same strategy
as in TM2:
    #resizing the image to be at least 224x224 and then
cropping from the center
    size = (224, 224)
    image = ImageOps.fit(image, size, Image.ANTIALIAS)

    #turn the image into a numpy array
    image_array = np.asarray(image)
    # Normalize the image
    normalized_image_array =
(image_array.astype(np.float32) / 127.0) - 1
    # Load the image into the array
    data[0] = normalized_image_array

    # run the inference
    prediction = model.predict(data)

    #get the 1D array
    output = prediction[0]
    #assign default value for max confidence
    max_index = 0
    max_confidence = output[0]
    #find the maximum confidence and its index
    for i in range(1, len(output)):
        if max_confidence < output[i]:
            max_confidence = output[i]
            max_index = i
    print(max_index, max_confidence)

    file = open("labels.txt",encoding="utf8")
    data = file.read().split("\n")
    print("AI Result: ", data[max_index])
    client.publish("ai", data[max_index])

client =
MQTTClient("BBC_IoT","aio_QYth850SuWaXh20PWdD20n0W5XSN")
client.connect()
client.loop_background()
while True:
    time.sleep(5)
    image_capture()
```