Multimedia
(Lab10)

Spring, 2020

Department of Software

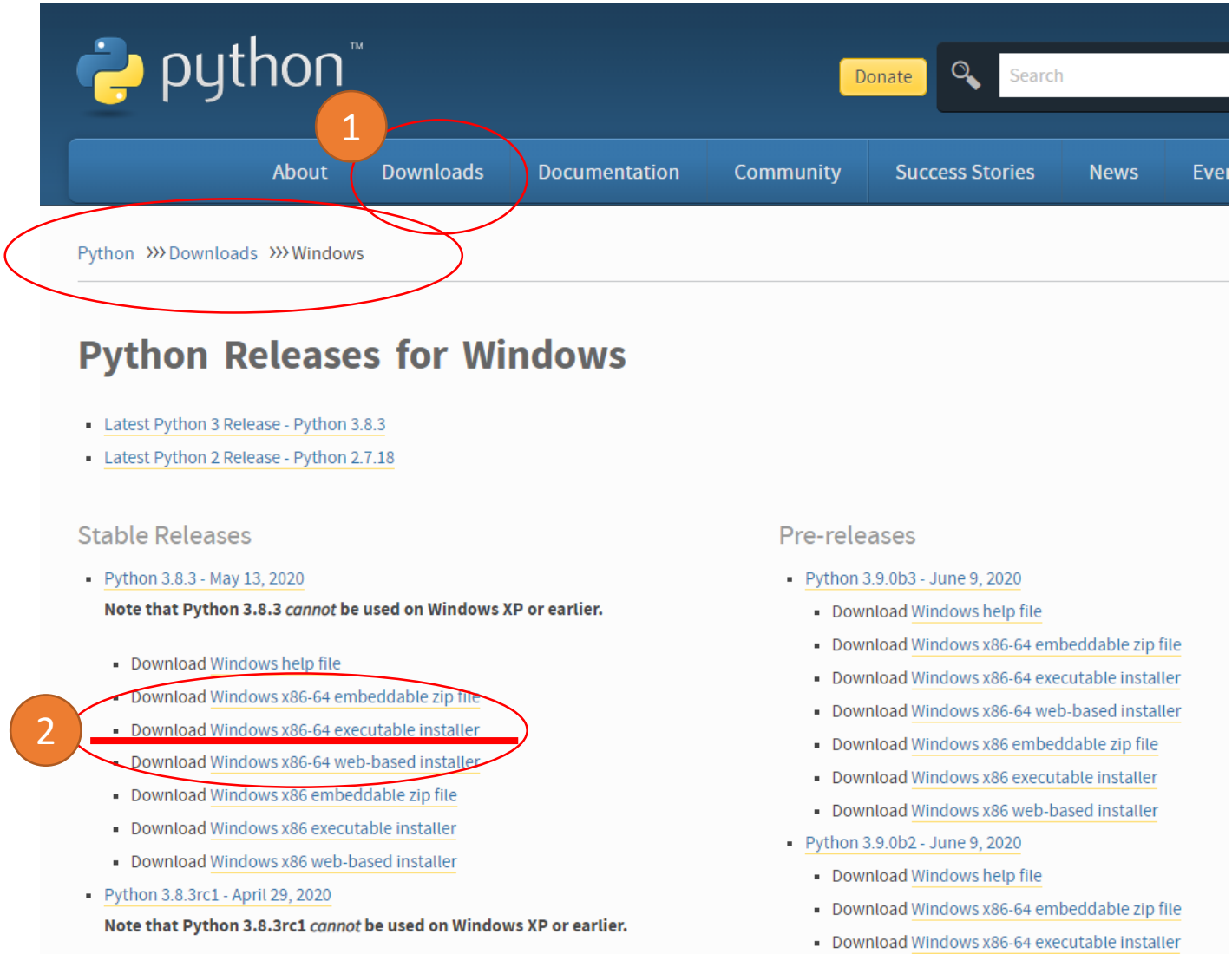
Yong Ju Jung (정용주)

미래창조과학부 SW중심대학
가천대학교 소프트웨어학과
Gachon University Department of Software

Lab 10-1
Local feature detection & matching

Install Python

Go to python.org



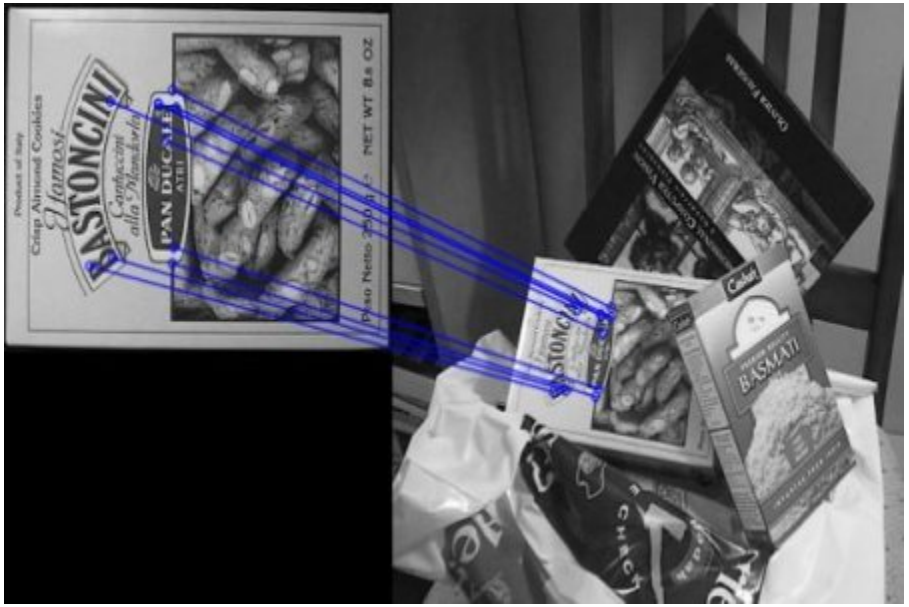
Install Python & OpenCV

- Install Python packages
  - Python 3.x (3.4+)
  - **Numpy** package (pip install numpy)
  - **Matplotlib** (pip install matplotlib)
  - **Jypyter lab** (pip install jupyterlab)
- Install OpenCV for Python
  - pip install opencv-python
  - (*optional*) pip install opencv-contrib-python

  - test in command window
    - >> python
    - >> import cv2
    - >> print(cv2.getVersionString())

ORB feature detection and matching

- Here, we will see a simple example on how to match features between two images.
  - https://docs.opencv.org/master/d1/d89/tutorial_py_orb.html
  - https://docs.opencv.org/master/dc/dc3/tutorial_py_matcher.html

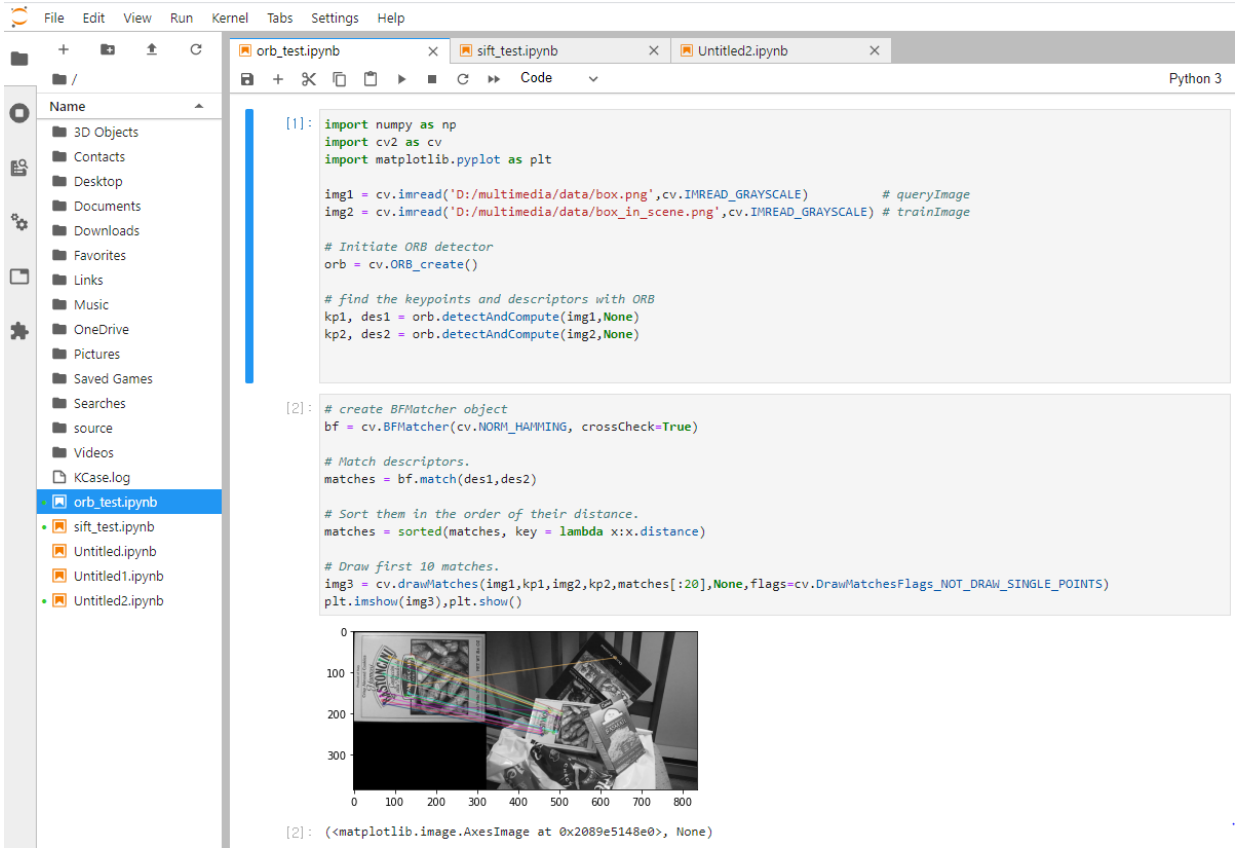


Rublee, E., Rabaud, V., Konolige, K., and Bradski, G., "ORB: an efficient alternative to SIFT or SURF," in ICCV 2011

https://jupyter.org/

Run your code in JupyterLab

- Run jupyter lab
  - Type in command window >> **jupyter lab**
  - It will automatically open your web browser



ORB feature detector

- ORB: **O**riented FAST and **R**otated **B**RIEF
  - Feature detection: Oriented FAST algorithm
  - Feature descriptor: Rotated BRIEF algorithm
    - It produces binary strings as feature description.

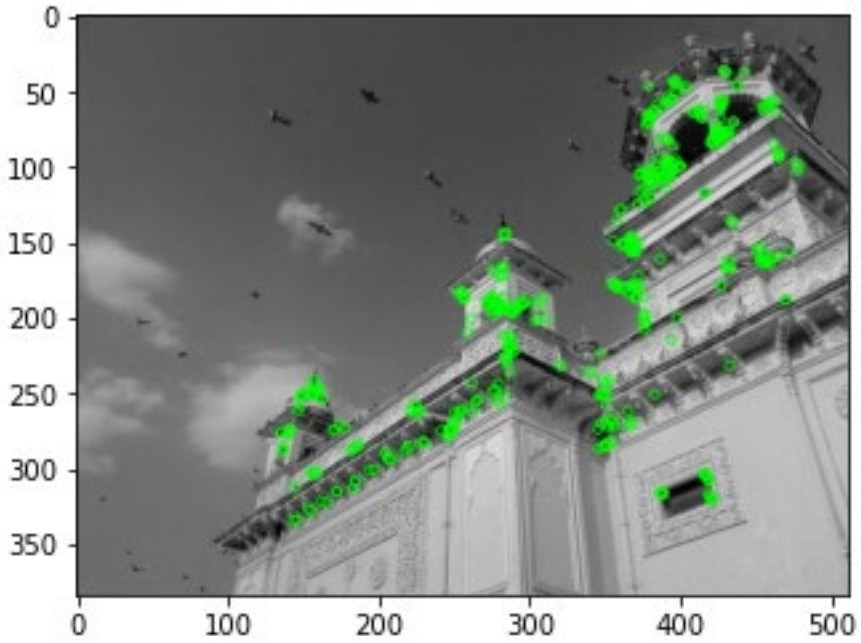
[1]:
```
import numpy as np
import cv2 as cv
from matplotlib import pyplot as plt
```

[2]:
```
img = cv.imread('D:/multimedia/data/box.png',0)

# Initiate ORB detector
orb = cv.ORB_create()
# find the keypoints with ORB
kp = orb.detect(img,None)
# compute the descriptors with ORB
kp, des = orb.compute(img, kp)
```

- Draw keypoints

[5]:
```
# draw only keypoints location,not size and orientation
img2 = cv.drawKeypoints(img, kp, None, color=(0,255,0), flags=0)
plt.imshow(img2), plt.show()
```



Feature Matching between Two Images

- We are using ORB descriptors to match features. So let's start with loading images, finding descriptors etc.

```
import numpy as np
import cv2 as cv
import matplotlib.pyplot as plt

img1 = cv.imread('box.png',cv.IMREAD_GRAYSCALE) # queryImage
img2 = cv.imread('box_in_scene.png',cv.IMREAD_GRAYSCALE) # trainImage

# Initiate ORB detector
orb = cv.ORB_create()

# find the keypoints and descriptors with ORB
kp1, des1 = orb.detectAndCompute(img1,None)
kp2, des2 = orb.detectAndCompute(img2,None)
```

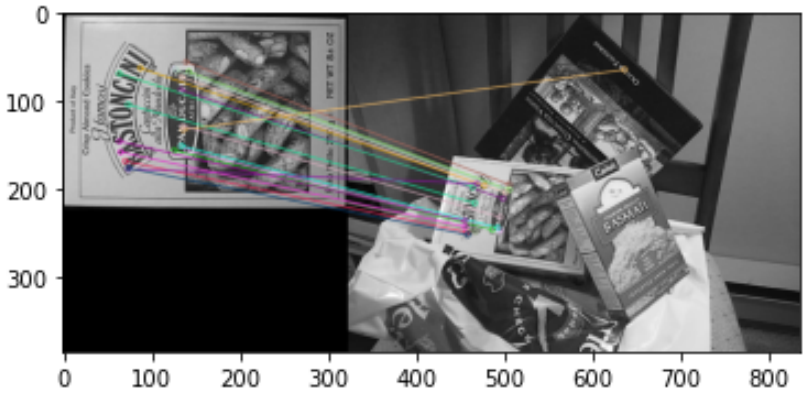
Brute-Force Matching with ORB Descriptors

- Use Hamming distance
  - Next we create a BFMatcher object with distance measurement **cv.NORM_HAMMING** (since we are using the binary strings).
- Then we use Matcher.match() method to get the best matches in two images.
- We sort them in ascending order of their distances so that best matches (with low distance) come to front.

```
# create BFMatcher object
bf = cv.BFMatcher(cv.NORM_HAMMING, crossCheck=True)
# Match descriptors.
matches = bf.match(des1,des2)
# Sort them in the order of their distance.
matches = sorted(matches, key = lambda x:x.distance)
```

- Draw matched feature points between two images

```
# Draw first 10 matches.
img3 = cv.drawMatches(img1,kp1,img2,kp2,matches[:20],None,flags=cv.DrawMatchesFlags_NOT_DRAW_SINGLE_POINTS)
plt.imshow(img3),plt.show()
```



SIFT feature

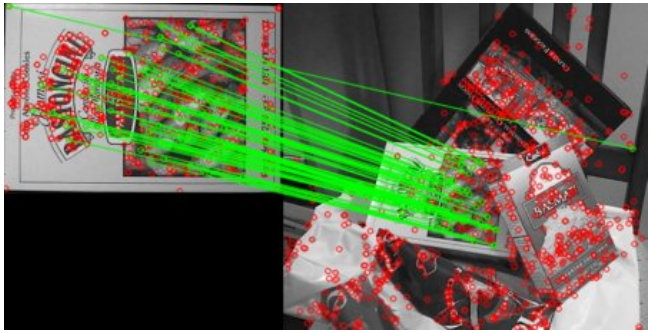
* Note: This algorithm is patented and is excluded in the recent OpenCV versions. Rebuild the OpenCV by yourself.
Set OPENCV_ENABLE_NONFREE CMake option and rebuild the library in function 'cv::xfeatures2d::SIFT::create'

- OpenCV library for Python

```
import numpy as np
import cv2 as cv
from matplotlib import pyplot as plt
img1 = cv.imread('box.png',0) # queryImage
img2 = cv.imread('box_in_scene.png',0) # trainImage
# Initiate SIFT detector
sift = cv.xfeatures2d.SIFT_create()
# find the keypoints and descriptors with SIFT
kp1, des1 = sift.detectAndCompute(img1,None)
kp2, des2 = sift.detectAndCompute(img2,None)

# FLANN parameters (for k-d Tree)
FLANN_INDEX_KDTREE = 1
index_params = dict(algorithm = FLANN_INDEX_KDTREE, trees = 5)
search_params = dict(checks=50) # or pass empty dictionary
flann = cv.FlannBasedMatcher(index_params,search_params)
matches = flann.knnMatch(des1,des2,k=2)  # extract two best matches
# Need to draw only good matches, so create a mask
matchesMask = [[0,0] for i in xrange(len(matches))]

# NNDR ratio test as per Lowe's paper
for i,(m,n) in enumerate(matches):
      if m.distance < 0.7*n.distance:
            matchesMask[i]=[1,0]
draw_params = dict(matchColor = (0,255,0),singlePointColor = (255,0,0),
           matchesMask = matchesMask,
flags = 0)
img3 = cv.drawMatchesKnn(img1,kp1,img2,kp2,matches,None,**draw_params)
plt.imshow(img3,),plt.show()
```



https://docs.opencv.org/3.4/dc/dc3/tutorial_py_matcher.html

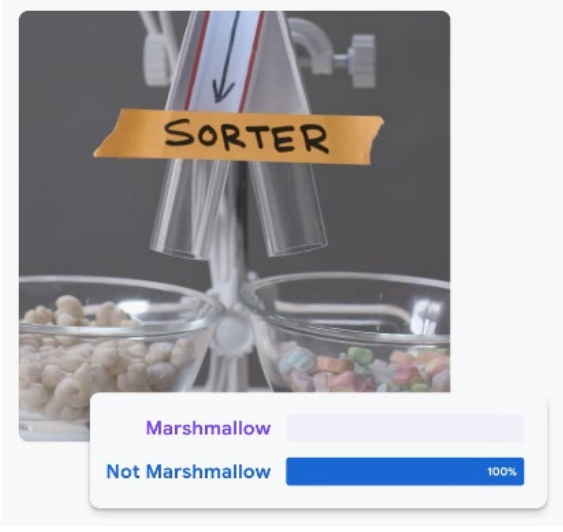
Lab 10-2:
Build your deep model using “Teachable Machine”

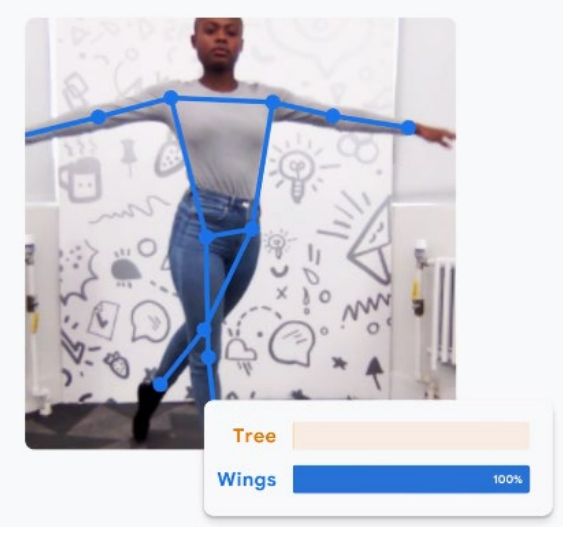
Teachable Machine

Google

- **Train a computer to recognize your own images, sounds, & poses.**
- A fast, easy way to create machine learning models for your sites, apps, and more – no expertise or coding required.

https://teachablemachine.withgoogle.com/





How do I use it?

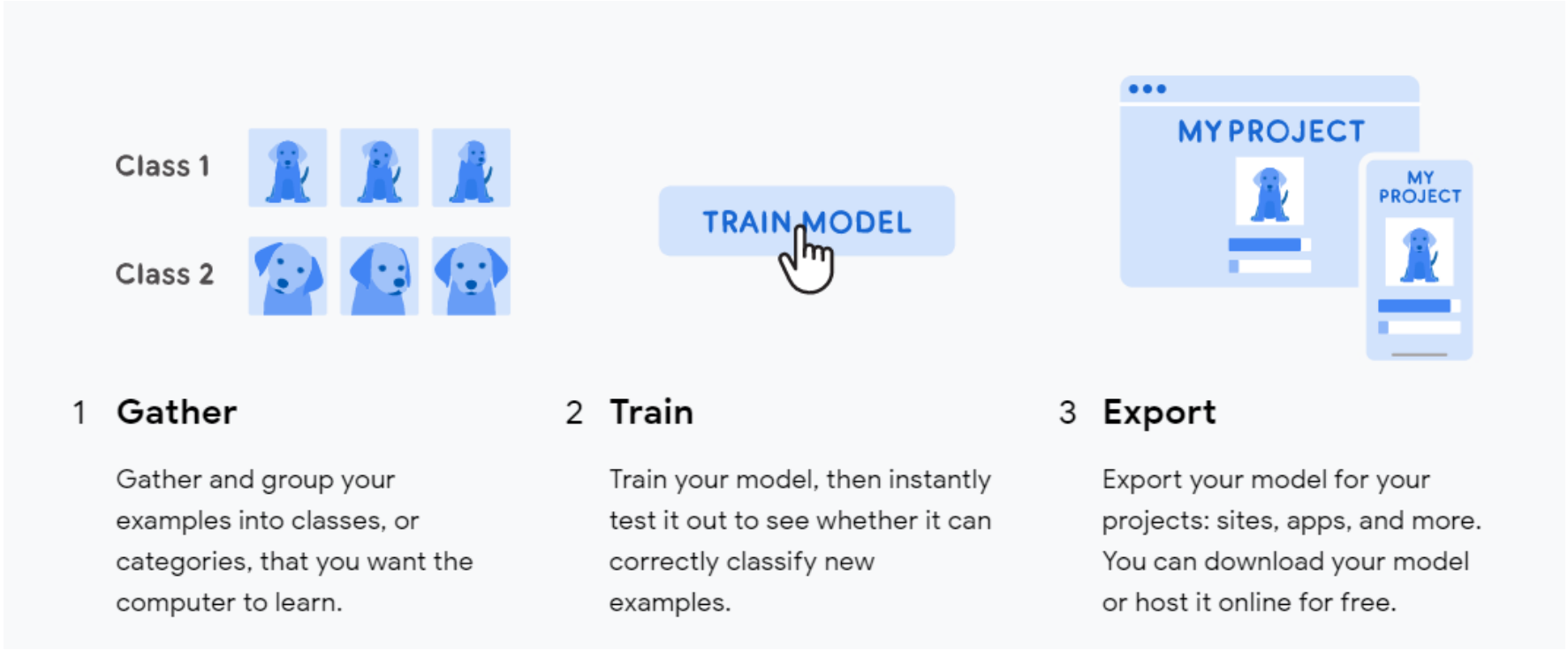
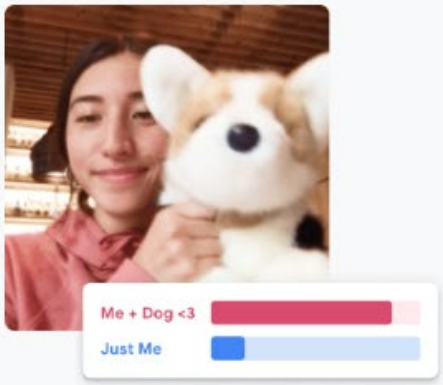


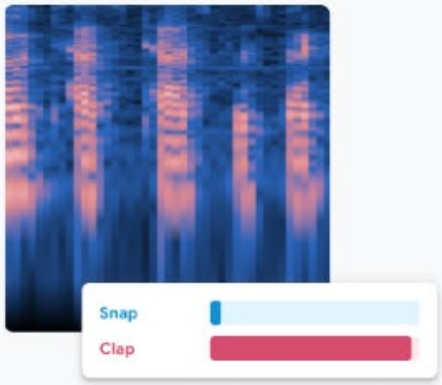
Figure from https://teachablemachine.withgoogle.com/

What can I use to teach it?



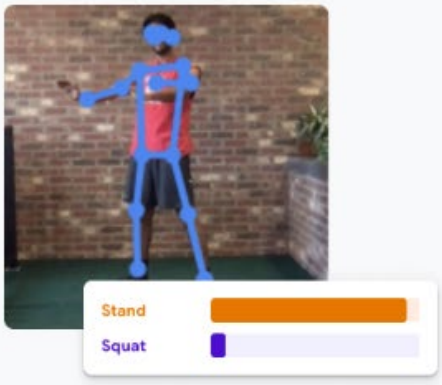
Images

Teach a model to classify images using files or your webcam.



Sounds

Teach a model to classify audio by recording short sound samples. (WAV/MP3/etc file support coming soon.)



Poses

Teach a model to classify body positions using files or striking poses in your webcam.

[Figure from https://teachablemachine.withgoogle.com/](https://teachablemachine.withgoogle.com/)

Build your model for image classification