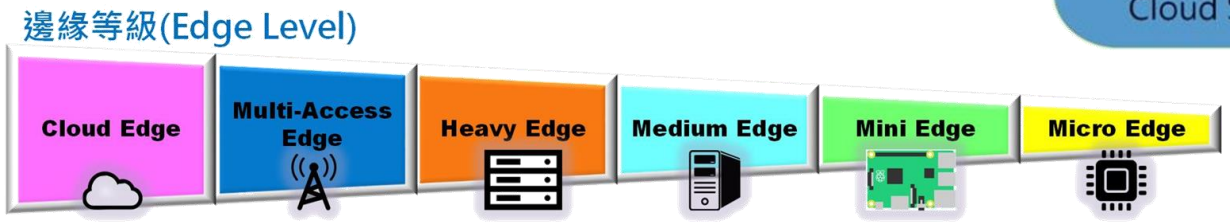
國立臺灣科技大學
NATIONAL TAIWAN UNIVERSITY OF SCIENCE AND TECHNOLOGY

Department of Computer Science & Information Engineering
資 訊 工 程 系

人工智慧與邊緣運算實務

Google Colab 進階應用

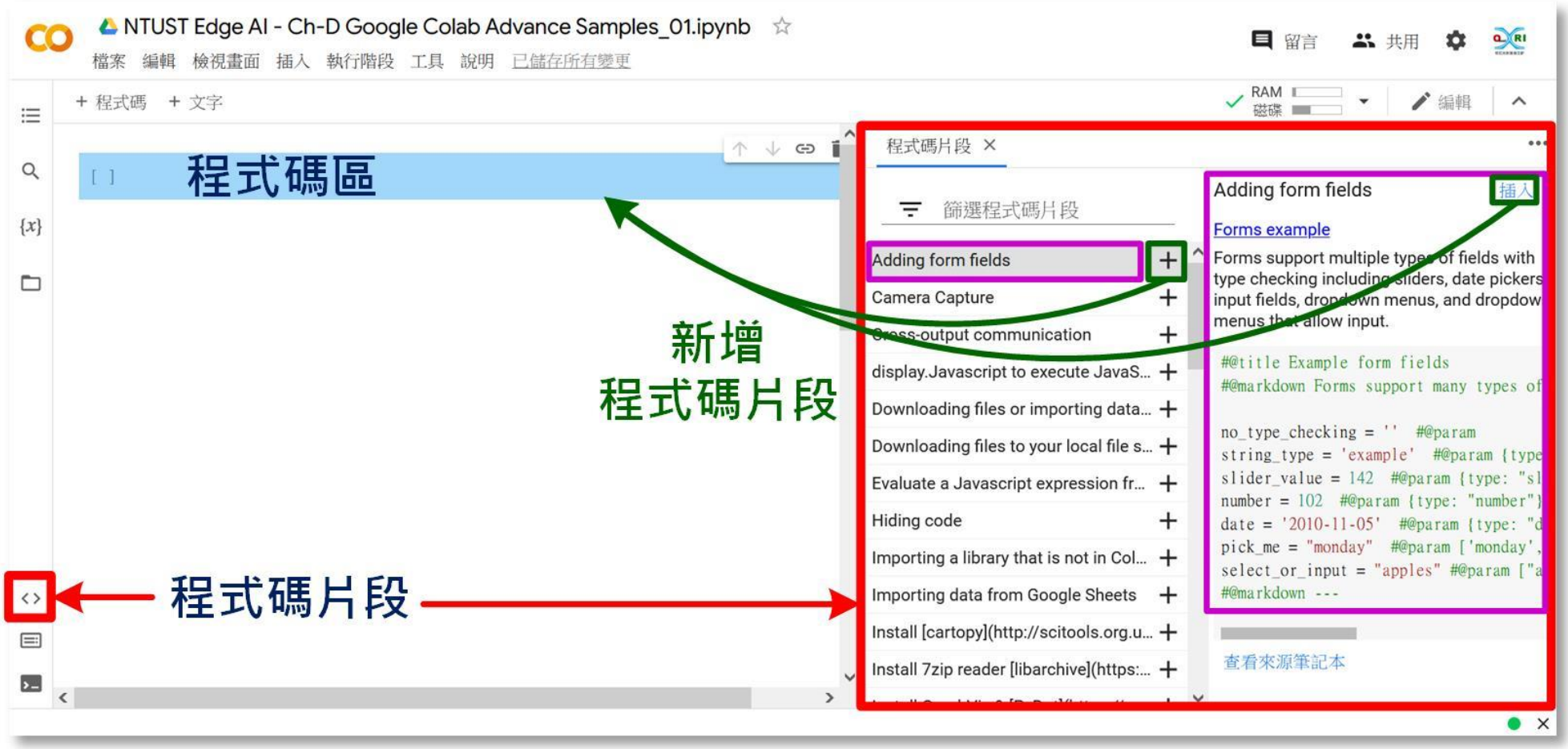


資訊工程系 許哲豪 助理教授

簡報大綱

- **Colab進階應用**
  - 程式碼片段
  - 表單控制項
  - 從攝影機取像
  - 虛擬機檔案傳輸
  - 雲端硬碟檔案傳輸
  - 顯示OpenCV影像及影片
  - 運行自定義影像處理
  - 資料可視化
  - 從Github運行ipynb

Google Colab 新增程式碼片段



程式碼片段範例 (1 of 3)

- **Adding form fields**
- **Camera Capture**
- Cross-output communication
- display.Javascript to execute JavaScript from Python
- Downloading files or importing data from Google Drive
- Downloading files to your local file system
- Evaluate a Javascript expression from Python with eval_js
- Hiding code
- Importing a library that is not in Colaboratory
- Importing data from Google Sheets
- Install [cartopy]
- Install 7zip reader [libarchive]
- Install GraphViz & [PyDot]
- Javascript to Python communication
- Jupyter Comms
- Jupyter Widgets

程式碼片段範例 (2 of 3)

- **Listing files in Google Drive Mounting Google Drive in your VM**
- Open files from GCS with gsutil
- Open files from GCS with the Cloud Storage Python API
- Open files from GitHub
- **Open files from Google Drive**
- Open files from your local file system
- Output Handling
- Pandas: display dataframes as interactive tables
- Pausing output processing
- **Saving data to Google Drive**
- Saving data to Google Sheets
- Saving data with gsutil
- Saving data with the Cloud Storage Python API
- Serving resources
- **Showing CV2 Images**
- Tagged Outputs
- Third-party Jupyter widgets

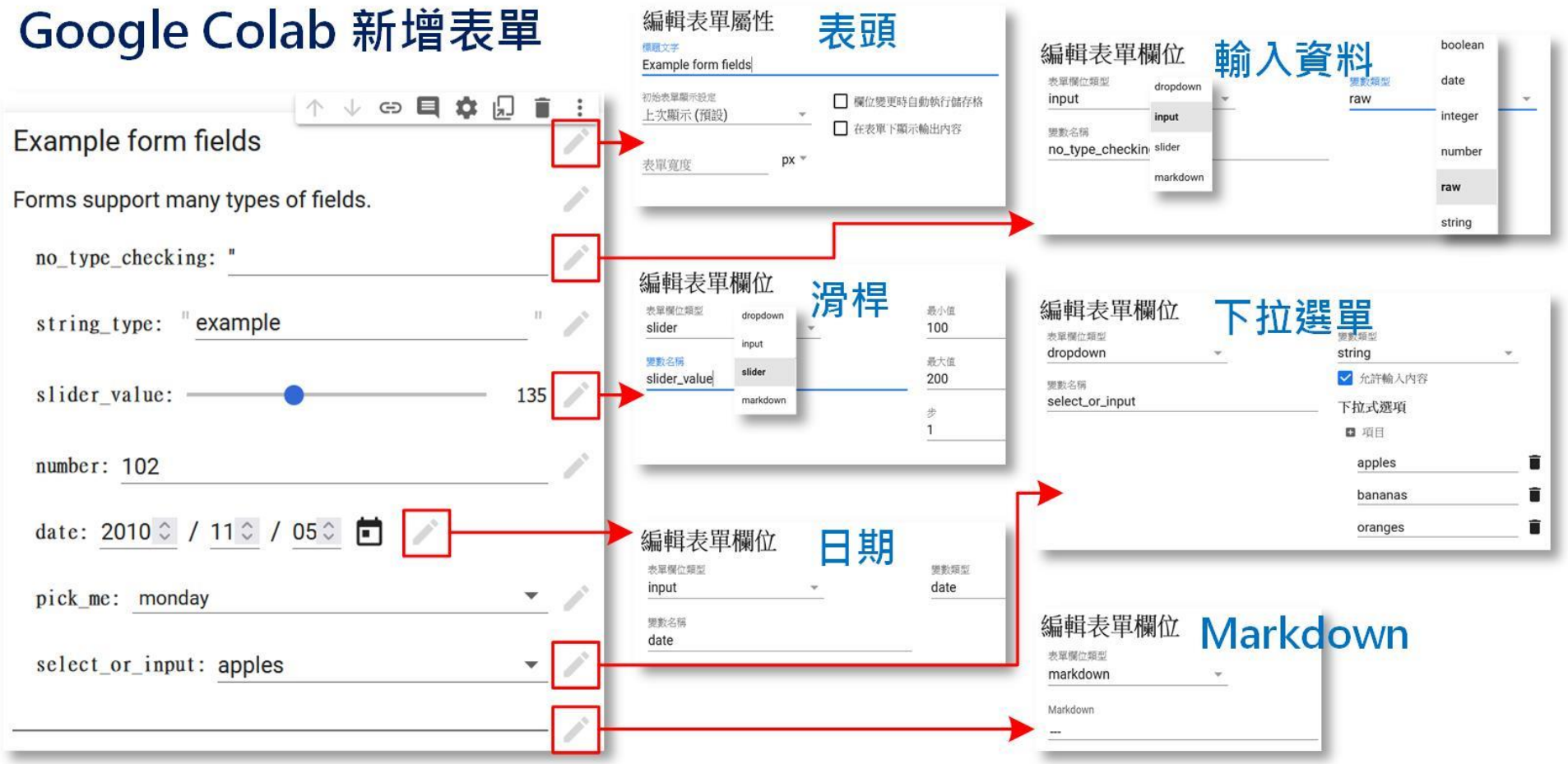
程式碼片段範例 (3 of 3)

- Using BigQuery with Cloud API
- Using BigQuery with Pandas API
- **Visualization:** Bar Plot in Altair
- Visualization: Histogram in Altair
- Visualization: Interactive Brushing in Altair
- Visualization: Interactive Scatter Plot in Altair
- Visualization: Linked Brushing in Altair
- Visualization: Linked Scatter-Plot and Histogram in Altair
- Visualization: Scatter Plot with Rolling Mean in Altair
- Visualization: Stacked Histogram in Altair
- Visualization: Time Series Line Plot in Altair

新增表單控制項 (1 of 2)

```
#@title Example form fields

#@markdown Forms support many types of fields.

no_type_checking = ''  #@param
string_type = 'example'  #@param {type: "string"}
slider_value = 135  #@param {type: "slider", min: 100, max: 200}
number = 102  #@param {type: "number"}
date = '2010-11-05'  #@param {type: "date"}
pick_me = "monday"  #@param ['monday', 'tuesday', 'wednesday', 'thursday']
select_or_input = "apples" #@param ["apples", "bananas", "oranges"] {allow-input: true}

#@markdown ---
```

新增表單控制項(2 of 2)

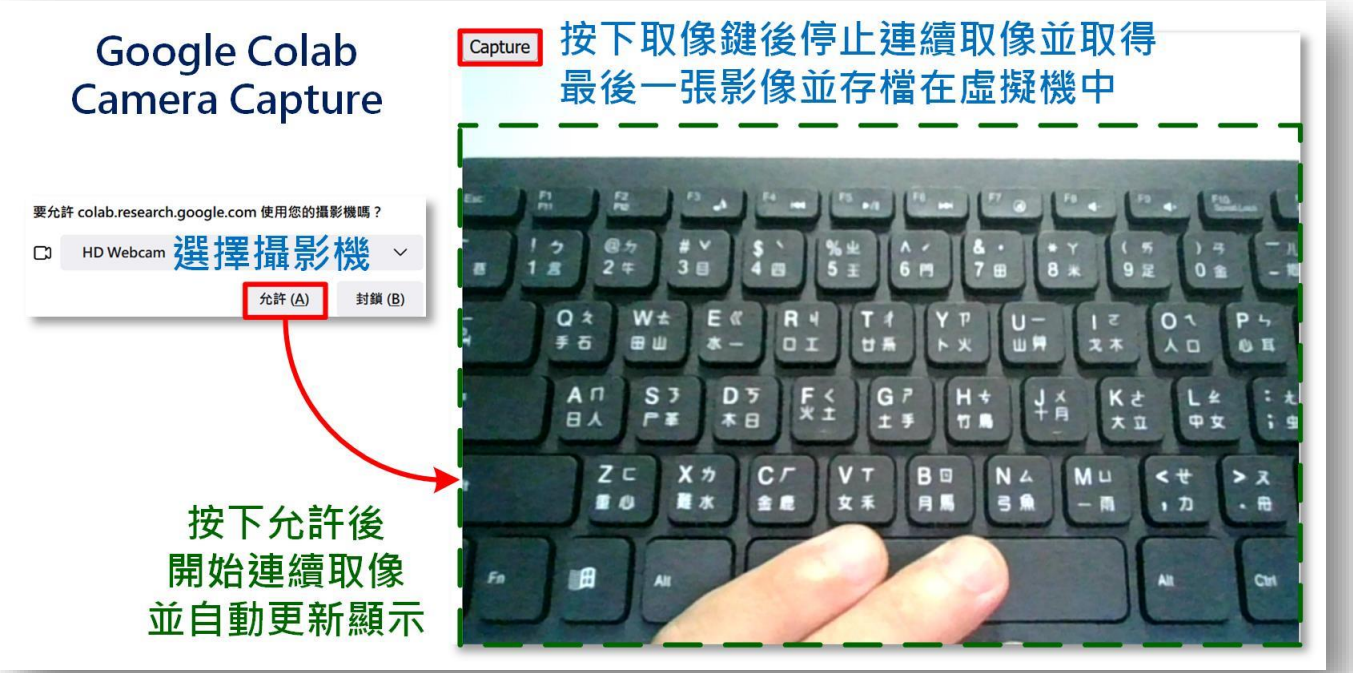


從攝影機取像 (1 of 2)

```
# 定義 Java 取像函式
from IPython.display import display, Javascript
from google.colab.output import eval_js
from base64 import b64decode

def take_photo(filename='photo.jpg', quality=0.8):
  js = Javascript('''
    async function takePhoto(quality) {
      const div = document.createElement('div');
      const capture = document.createElement('button');
      capture.textContent = 'Capture';
      div.appendChild(capture);

      const video = document.createElement('video');
      video.style.display = 'block';
      const stream = await navigator.mediaDevices.getUserMedia({video: true});

      document.body.appendChild(div);
      div.appendChild(video);
      video.srcObject = stream;
      await video.play();

      // Resize the output to fit the video element.
      google.colab.output.setIframeHeight(document.documentElement.scrollHeight, true);

      // Wait for Capture to be clicked.
      await new Promise((resolve) => capture.onclick = resolve);

      const canvas = document.createElement('canvas');
      canvas.width = video.videoWidth;
      canvas.height = video.videoHeight;
      canvas.getContext('2d').drawImage(video, 0, 0);
      stream.getVideoTracks()[0].stop();
      div.remove();
      return canvas.toDataURL('image/jpeg', quality);
    }
    ''')
  display(js)
  data = eval_js('takePhoto({})'.format(quality))
  binary = b64decode(data.split(',')[1])
  with open(filename, 'wb') as f:
    f.write(binary)
  return filename
```

從攝影機取像 (2 of 2)

```
# 開始取像，當按下Capture時將影像存於虛擬機上
from IPython.display import Image
try:
  filename = take_photo()
  print('Saved to {}'.format(filename))

  # Show the image which was just taken.
  display(Image(filename))
except Exception as err:
  # Errors will be thrown if the user does not have a webcam or if they do not
  # grant the page permission to access it.
  print(str(err))
```



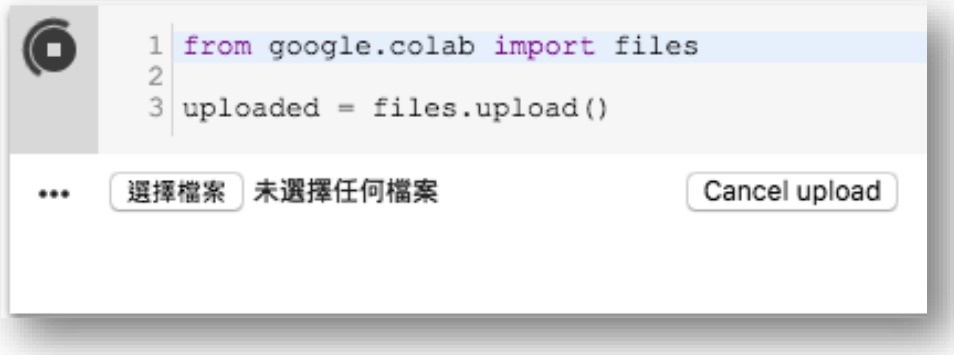
Colab虛擬機與本地端下載/上傳

➢從Colab下載檔案到本地端

```
# 導入檔案處理函式庫
from google.colab import files

# 開啟文字檔，寫入內容。
with open('example.txt', 'w') as f:
  f.write('some content’  )
# 下載檔案到本地端
files.download('example.txt')
```

➢從本地端上傳檔案到Colab

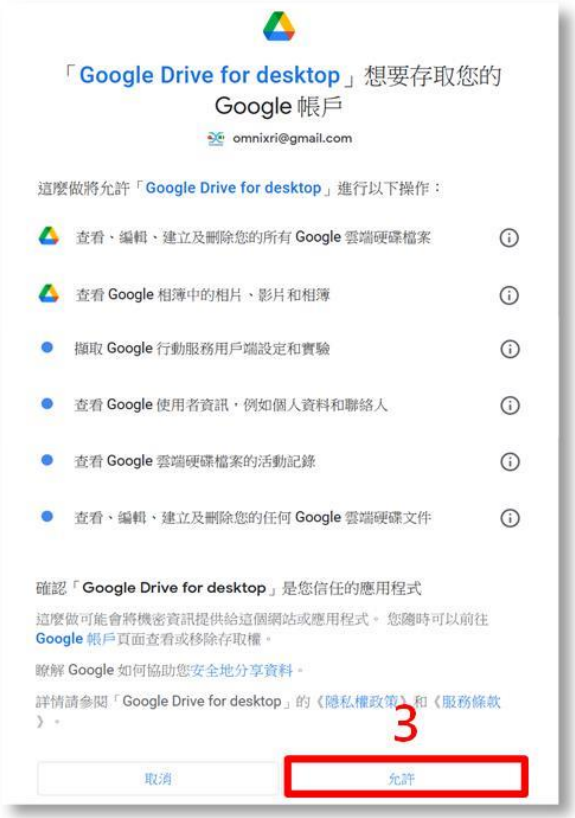
```
# 導入檔案處理函式庫
from google.colab import files

# 選擇本地端檔案上傳到虛擬機
uploaded = files.upload()
```

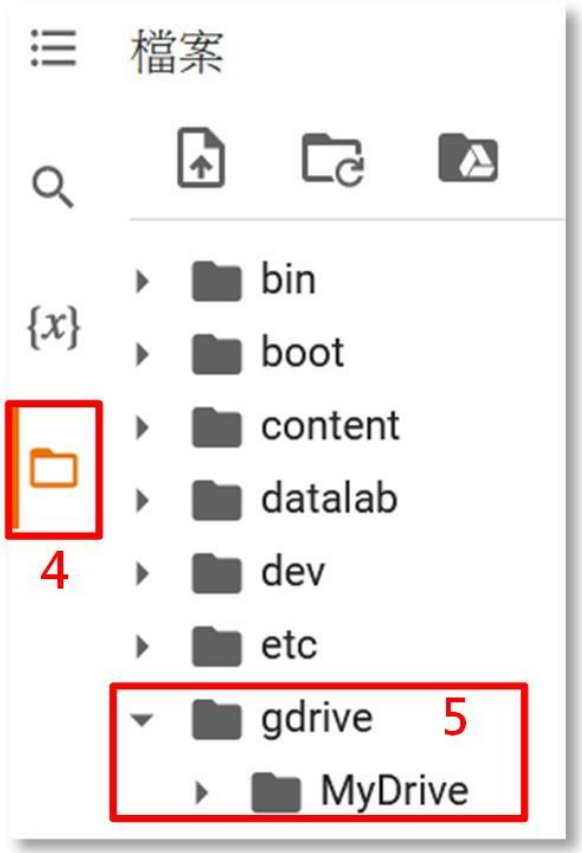


掛載Google雲端硬碟到Colab VM

Google Drive 掛載

```
from google.colab import drive
drive.mount('/gdrive')
```





更多本機和雲端檔案存取 https://colab.research.google.com/notebooks/io.ipynb

以Colab函數顯示OpenCV圖像

```
from google.colab.patches import cv2_imshow # 導入Colab函式庫
!curl -
o logo.png https://colab.research.google.com/img/colab_favicon_256px.png
import cv2 # 導入OpenCV函式庫
img = cv2.imread( 'logo.png' , cv2.IMREAD_UNCHANGED) # 讀入圖檔
cv2_imshow(img) # 顯示OpenCV（BGR）格式影像
```

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  4534  100  4534    0     0   163k      0 --:--:-- --:--:-- --:--:--  163k
```

OpenCV原始
cv2.imshow()
在Colab下無法運行

直接顯示 OpenCV
BRG888 格式
不用轉換色彩空間

以IPython.display及PIL顯示

```
from PIL import Image # 導入PIL Image函式庫

from IPython.display import display, clear_output # 導入函式庫


img3 = cv2.imread('OmniXRI_Logo.jpg') #讀入影像檔

img4 = cv2.cvtColor(img1, cv2.COLOR_BGR2RGB) # 將影像色彩空間從BGR888 轉成 RGB888


display(Image.fromarray(img4)) # 顯示影像
```

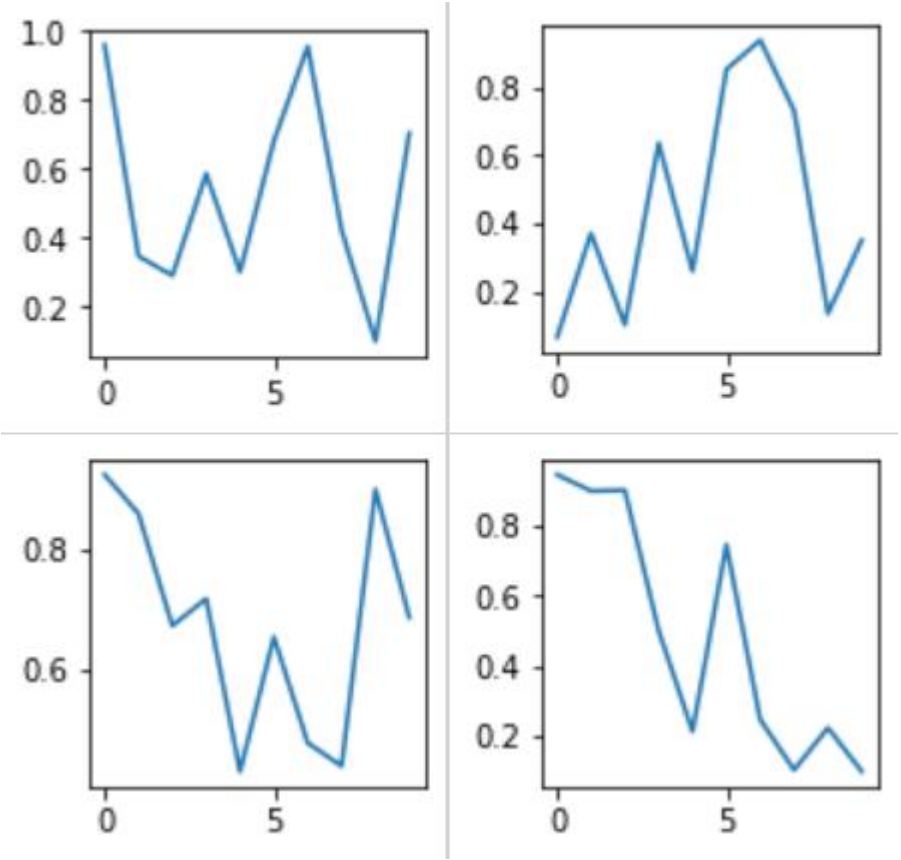
以matplotlib方式顯示OpenCV圖像

```
import cv2 # 導入OpenCV函式庫
import matplotlib.pyplot as plt # 導入matplotlib.pyplot函式庫

img1 = cv2.imread( 'gdriver/MyDrive/xxx.jpg' ) #讀入掛載之雲端硬碟中的影像檔或者先以!wget指令從網路上下載影像到虛擬機上再讀入

img2 = cv2.cvtColor(img1, cv2.COLOR_BGR2RGB) # 將影像色彩空間從 BGR888 轉成 RGB888

plt.axis("off") # 設定關閉XY軸刻尺
plt.imshow(img2) # 繪製單張影像
plt.show() # 顯示影像
```

Matplot配合Grid widgets繪圖

```
import numpy as np
import random
import time
from matplotlib import plt

# 設定Grid為2x2
Grid = widgets.Grid(2, 2)

For I in range(20):
  with grid.output_to(random.randint(0, 1
), random.randint(0, 1)):
    grid.clear_cell()
    plt.figure(figsize=(2, 2))
    plt.plot(np.random.random((10, 1)))
  time.sleep(0.5)
```

- 輸出結果



以IPython.display顯示影片

- 在Colab中無法以連續單張顯示的方法播放影片，可以使用IPython.display的HTML函式即可在線上顯示影片結果，以下範例以一個H.264(avc)格式的mp4檔案進行測試。

```
from IPython.display import HTML # 導入IPython.display HTML函式庫
from base64 import b64encode # 導入base64 baseencode函式庫

vs1 = open('gdriver/MyDrive/test/OmniXRI_Logo.mp4','rb').read() # 開啟並讀取mp4格式影片檔
data_url = "data:video/mp4;base64," + b64encode(vs1).decode() # 設定顯示內容格式
HTML(f'') # 將影片顯示於視窗上
```

以ffmpeg顯示OpenCV產出影片

- 一般OpenCV是以VideoCapture()來讀取及輸出影片檔案，但在Colab上無法直接顯示，只能把輸出的檔案存回雲端硬碟上，再以其它工具開啟顯示。
- OpenCV目前產出的mp4格式為mpv4格式，無法在Colab上顯示，所以要用ffmpeg將mpv4格式的mp4檔案轉成avc格式的mp4檔案，才能直接顯示。

參考範例：

https://github.com/OmniXRI/Colab_OpenCV_Display/blob/main/Colab_OpenCV_Display.ipynb

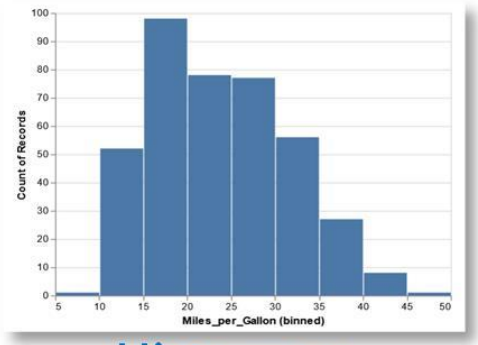
Colab使用Webcam運行自定義影像處理

- 目前Colab提供的「程式碼片段」範例僅可以連結本地端網路攝影機進行連續取像及顯示，且只能取得最後一張影像後，再以靜態方式（如OpenCV, PIL等）進行處理及顯示，無法於取像過程進行自定義影像處理。本範例提供如何解決執行自定義影像處理方案。

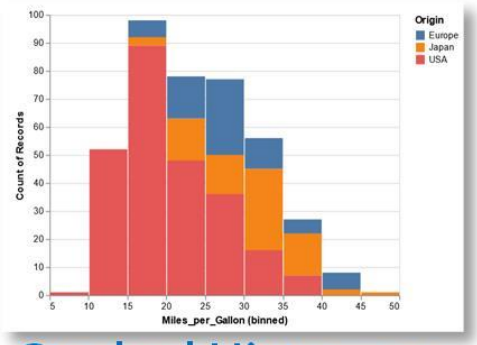
https://github.com/OmniXRI/Colab_Webcam_OpenCV

資料可視化 (Altair)

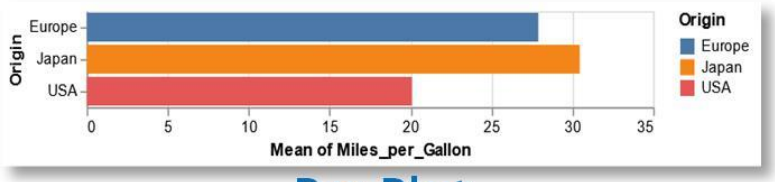
Google Colab - Visualization in Altair



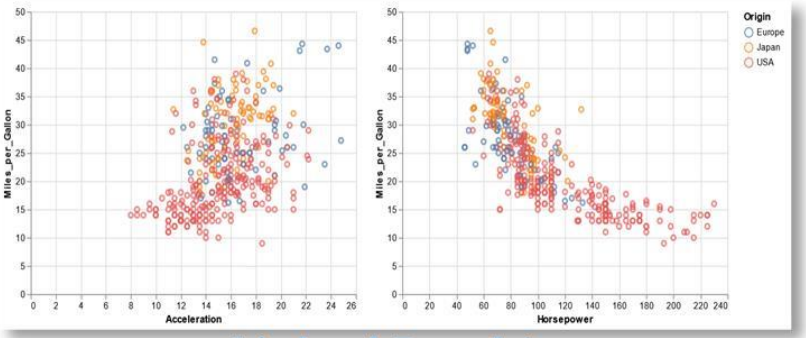
Histogram



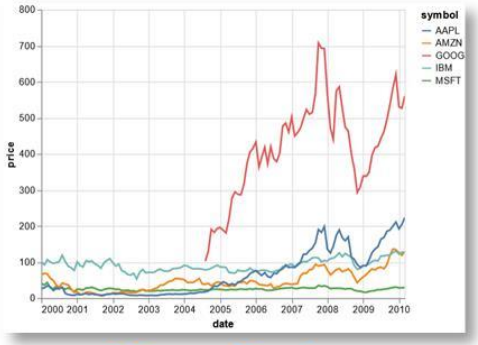
Stacked Histogram



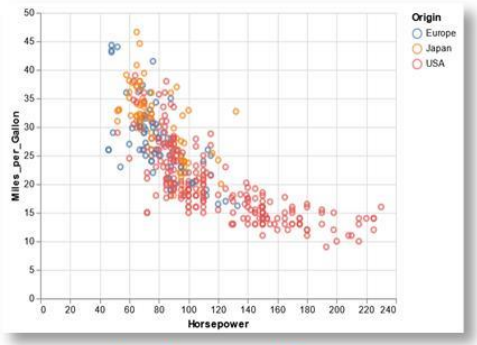
Bar Plot



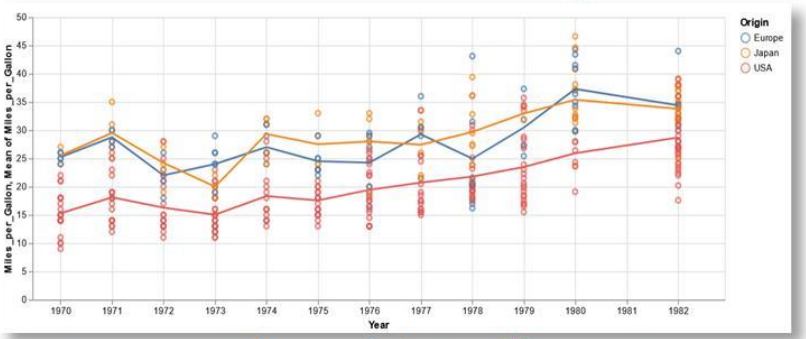
Linked Brushing



Time Series
Line Plot



Interactive
Brushing / Scatter



Scatter Plot with Rolling Mean

從Github運行ipynb

- 先登入個人 Google 帳號
- 假設 Github 中有一個 *.ipynb的程式

https://github.com/OmniXRI/colab_openvino/blob/master/Colab_OpenVINO_mobilenet_v1.ipynb

- 只要把路徑名稱中的「github.com」換成「colab.research.google.com/github」，其餘不變，即可直接以Colab Jupyter Notebook方式執行程式。
- 開啟後只能執行不能修改，建議可另存備份到自己的Google Drive，之後就能任意修改。

小結

- Google Colab提供了很多現成的範例（程式碼片段）方便使用者學習及開發。
- Colab支援 Java 和 Python 之間的溝通，方便透過網頁驅動遠端（雲端）和本地端的連結及互動。
- 提供基本操作元件，方便設計簡單人機互動介面。
- 可輕易整合Python版本的影像處理及繪圖函式庫，包括 OpenCV, PIL, Matplotlib等。
- 提供第三方資料可視化(Visualization)元件，方便顯示數值資料分佈。

參考文獻

- Google Colab官方說明文件

https://colab.research.google.com/

- NTUST Edge AI Ch 4.3.1 Google Colab基本介紹

https://omnixri.blogspot.com/p/ntust-edge-ai-ch4-3.html

- 許哲豪，如何在Colab上顯示雲端硬碟(Google Drive)上的影像和視頻

https://omnixri.blogspot.com/2020/12/colabgoogle-drive.html

- 許哲豪，如何在Google Colab上使用本地端Webcam即時運行OpenCV自定義影像處理函式

https://omnixri.blogspot.com/2022/09/google-colabwebcamopencv.html