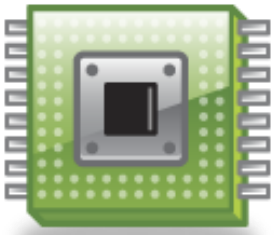
本課程為教育部109年度智慧聯網技術課程推廣計畫之補助課程
教材內容教育部智慧聯網居家照護之影像辨識和跌倒偵測應用開發模組教材改編



# TensorFlow Lite Based AI Edge Computing

謝東佑
國立中山大學電機系

Office: エEC-7038
07-5252000 Ext. 4114
tyhsieh@mail.ee.nsysu.edu.tw

# AI+Edge Computing四大應用場域

建築(building-scale)

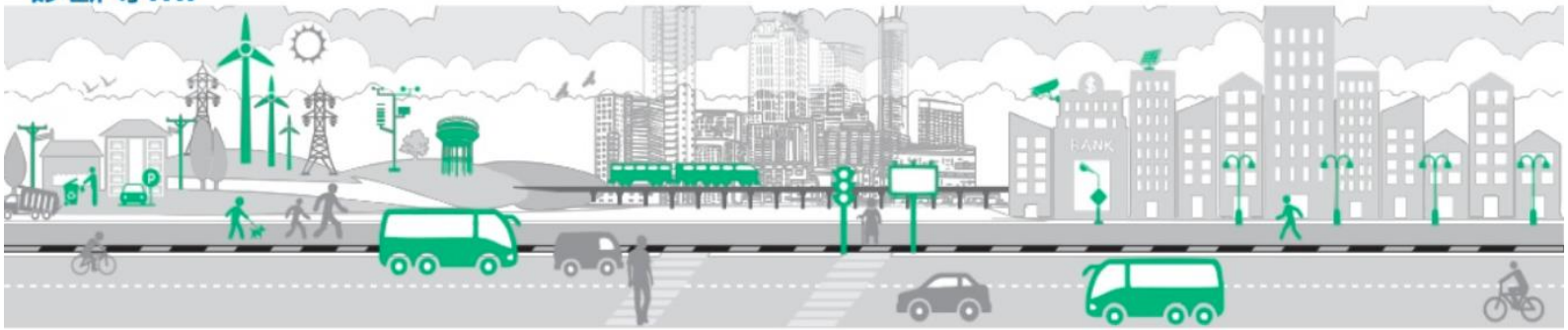
工廠/醫院內部影像辨識、機器人控制、機台數據分析、醫療診斷等....


家庭(room-scale)

家庭內如智慧音箱、智慧家電

個人(personal-scale)

手機、平板、穿戴裝置、AR/VR頭盔....



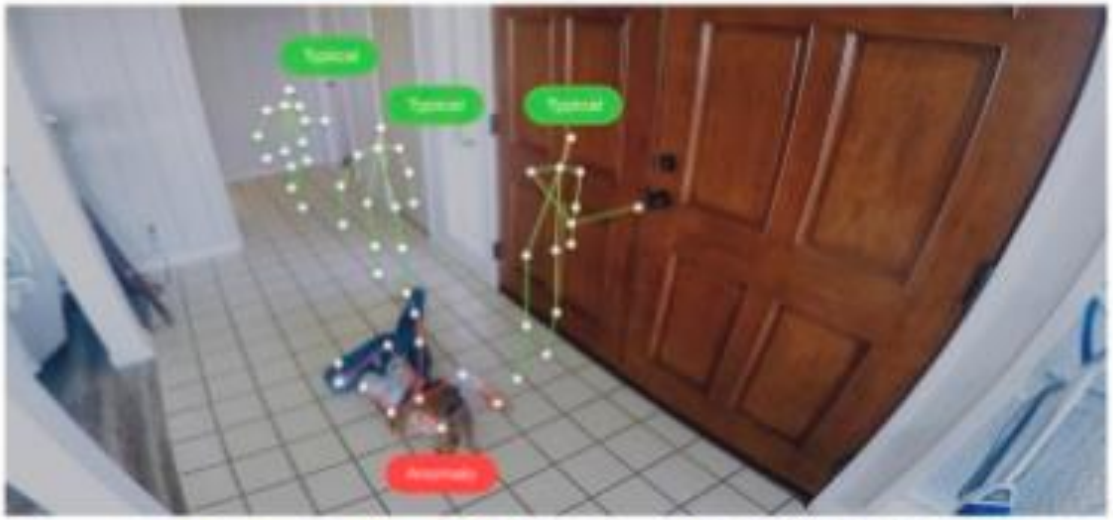
城市(city-scale)

自駕車、無人機、自走載具、街頭監控系統....

資料來源：工研院產科國際所 (2019/03)

# AI x 智慧健康：利用行為特徵與邊緣運算，兼顧居家安全及個人隱私

- 新創Cherry Labs提出智慧居家安全方案Cherry Home，安裝數個動作感測器與AI主機，主機與感測器之間則以 Wi-Fi連結
- 將AI分析與辨識能力置入裝置晶片中，透過邊緣運算能力可於主機中進行分析，為每位家庭成員建立行為模式，如行為模式異常即可能是外人或小偷
- 關於個資與隱私保護，系統演算法根據行為模式、身高等數據，建構出類似「火柴人」的「虛擬骨骼」骨架模型，即可具備判讀身份能力，過程中不涉及個資記錄與使用

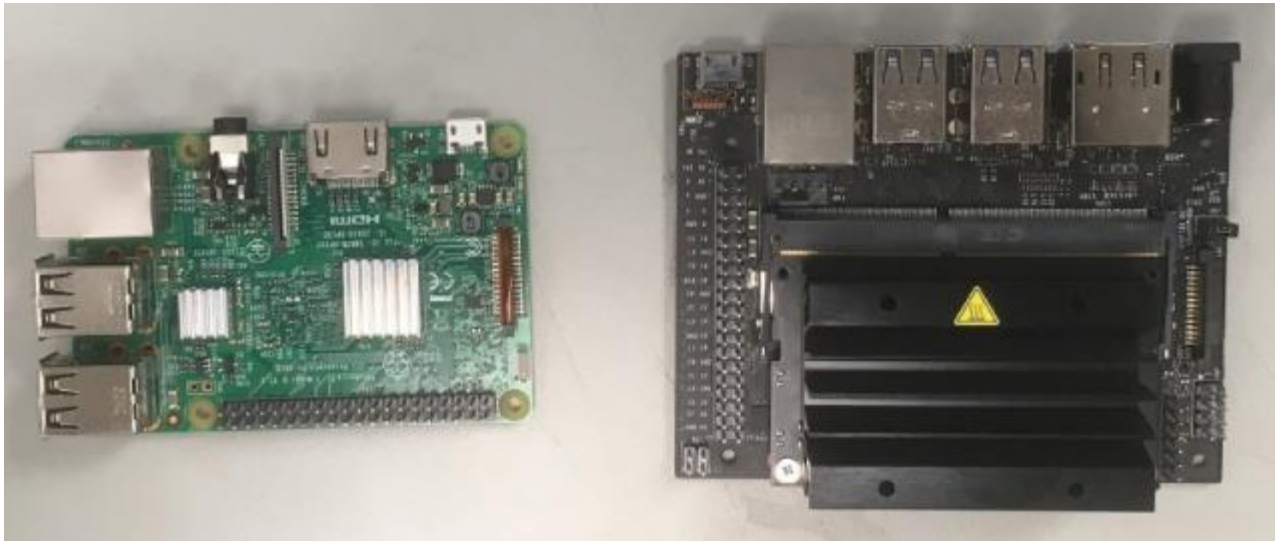


Cherry Home「虛擬骨骼」影像判斷技術

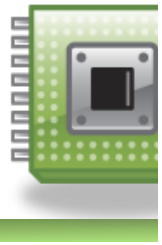
Walabot Home可放置於容易跌倒場所，如浴室

資料來源：工研院產科國際所

# Jetson Nano VS Raspberry Pi

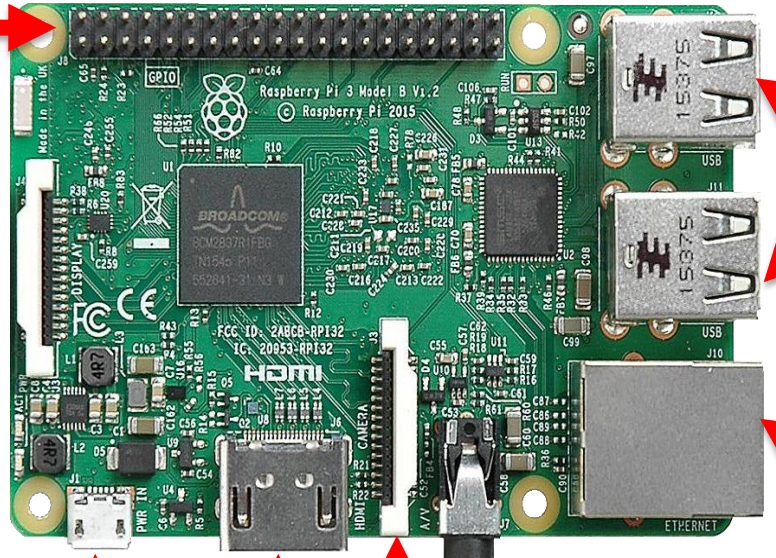


Source: https://chtseng.wordpress.com/2019/05/01/nvida-jetson-nano-初體驗：安裝與測試

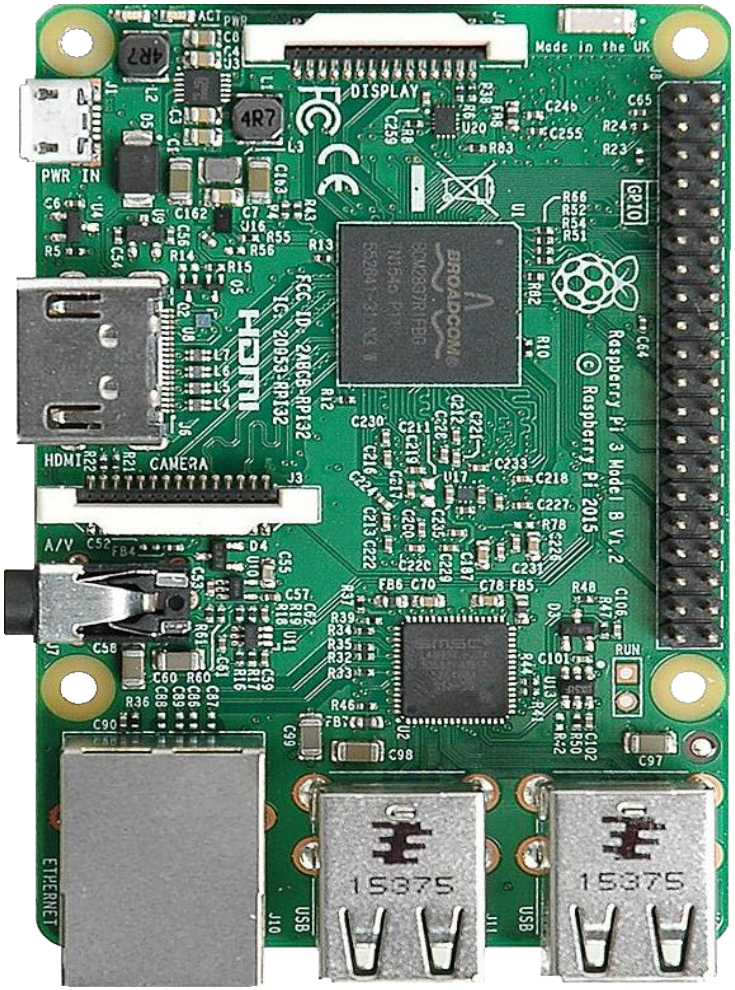


| | Jetson Nano | Raspberry Pi |
|---|---|---|
| CPU | 64-bit Quad-core ARM A57 (1.43 GHz) | 1.4 GHz 64-bit quad-core ARM Cortex-A53 |
| GPU | 128-Core Nvidia Maxwell | Broadcom VideoCore IV |
| RAM | 4GB DDR4 | 1GB DDR2 |
| WIFI | X | 802.11ac |
| Bluetooth | X | LE 4.2 |
| Ethernet | Gigbit | Gigbit (300Mbps max) |
| GPIO | 40 pin | 40 pin |
| USB | USB 2.0 x 3<br>USB 3.0 x 1 | USB 2.0 x 4 |
| Audio | X | Audio jack |
| Power | 5~10W | 400 mA (2.0W) |
| Price | $99 | $35 |
| 算力 | 472 Gflops | 24 Gflop |

# Raspberry Pi 3 model B



# General-Purpose Input/Output (GPIO)



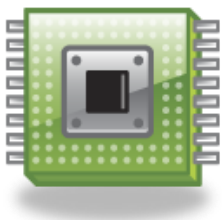

**Note that the physical PIN number is different from the GPIO number.**

Raspberry Pi 3 GPIO Header

| Pin# | NAME | NAME | Pin# |
|---|---|---|---|
| 01 | 3.3v DC Power | DC Power 5v | 02 |
| 03 | GPIO02 (SDA1 , I²C) | DC Power 5v | 04 |
| 05 | GPIO03 (SCL1 , I²C) | Ground | 06 |
| 07 | GPIO04 (GPIO_GCLK) | (TXD0) GPIO14 | 08 |
| 09 | Ground | (RXD0) GPIO15 | 10 |
| 11 | GPIO17 (GPIO_GEN0) | (GPIO_GEN1) GPIO18 | 12 |
| 13 | GPIO27 (GPIO_GEN2) | Ground | 14 |
| 15 | GPIO22 (GPIO_GEN3) | (GPIO_GEN4) GPIO23 | 16 |
| 17 | 3.3v DC Power | (GPIO_GEN5) GPIO24 | 18 |
| 19 | GPIO10 (SPI_MOSI) | Ground | 20 |
| 21 | GPIO09 (SPI_MISO) | (GPIO_GEN6) GPIO25 | 22 |
| 23 | GPIO11 (SPI_CLK) | (SPI_CE0_N) GPIO08 | 24 |
| 25 | Ground | (SPI_CE1_N) GPIO07 | 26 |
| 27 | ID_SD (I²C ID EEPROM) | (I²C ID EEPROM) ID_SC | 28 |
| 29 | GPIO05 | Ground | 30 |
| 31 | GPIO06 | GPIO12 | 32 |
| 33 | GPIO13 | Ground | 34 |
| 35 | GPIO19 | GPIO16 | 36 |
| 37 | GPIO26 | GPIO20 | 38 |
| 39 | Ground | GPIO21 | 40 |

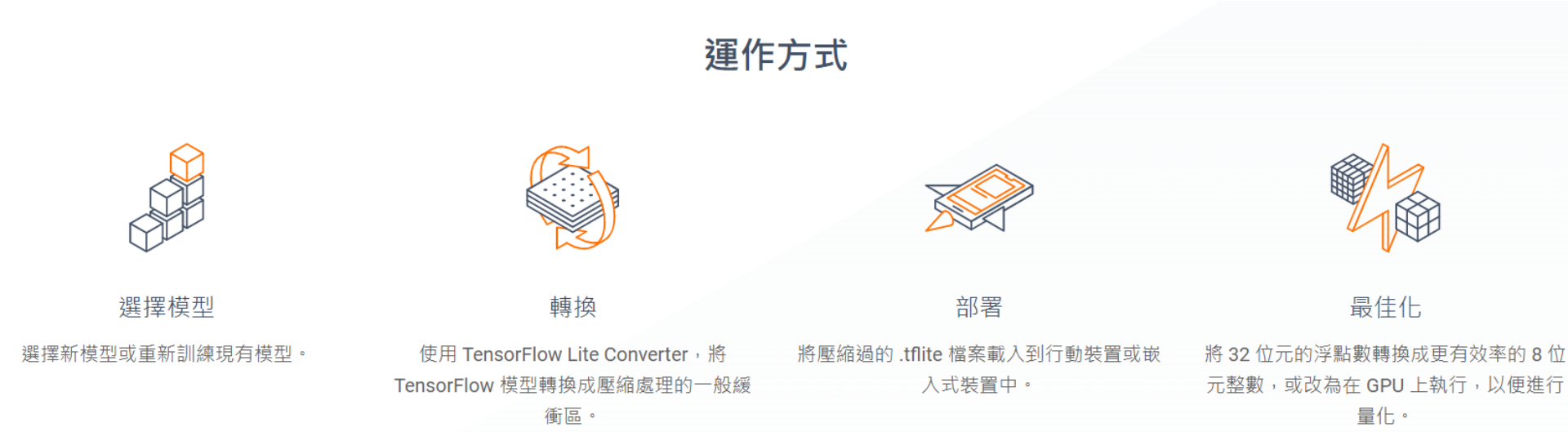
Rev. 2
29/02/2016

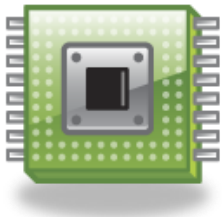
www.element14.com/RaspberryPi



# 在行動裝置和 IoT 裝置上部署機器學習模型

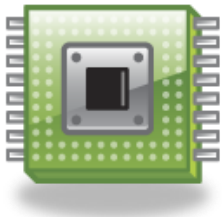
- **TensorFlow Lite**





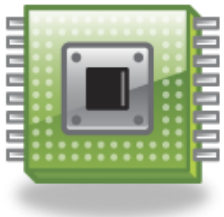
# TensorFlow Lite

- 讓開發人員能在裝置端輕鬆執行機器學習，不必透過伺服器來回傳送資料，可直接在網路的「邊緣」執行。對開發人員來說，在裝置端執行機器學習有助於改善以下項目：
  - 延遲情況：不必透過伺服器來回傳送資料
  - 隱私性：資料無須離開裝置
  - 連線：不需要網際網路連線
  - 耗電量：可節省網路連線的龐大耗電量



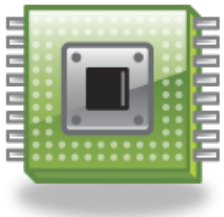
# TensorFlow Lite

- TensorFlow Lite 可用於多種裝置，從小型的微控制器到功能強大的手機都適用。
- TensorFlow Lite 由兩個主要元件組成：
  - TensorFlow Lite 解譯器：可在許多不同類型的硬體上 (包括手機、嵌入式 Linux 裝置和微控制器) 執行經過特別最佳化的模型。
  - TensorFlow Lite 轉換工具：可將 TensorFlow 模型轉換為方便解譯器使用的格式，並且可透過最佳化來降低二進位檔的大小及提高效能。



# TensorFlow Lite 檔案大小

- 系統支援的超過 125 個運算子全都連結時，TensorFlow Lite 二進位檔的大小約為 1 MB (32 位元 ARM 版本)
- 僅使用支援常見圖片分類模型 InceptionV3 和 MobileNet 所需的運算子時，則會小於 300 KB。

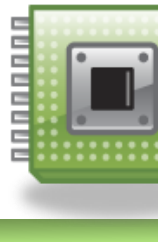


# TensorFlow Lite 入門

## 在行動裝置和 IoT 裝置上部署機器學習模型

TensorFlow Lite 是一種開放原始碼深度學習架構，可在裝置端執行推論。

| 參閱指南 | 參閱範例 | 查看模型 |
| --- | --- | --- |
| 說明 TensorFlow Lite 概念與元件的指南。 | 探索 TensorFlow Lite Android 版和 iOS 版應用程式。 | 輕鬆部署預先訓練模型。 |



## 開始使用

請前往開始使用頁面，瞭解如何在行動裝置上開始使用 TensorFlow Lite。如要將 TensorFlow Lite 模型部署至微控制器，請前往微控制器頁面。

## 主要功能

- 解譯器已針對裝置端的機器學習進行調整：支援一系列針對裝置端應用程式進行最佳化的核心運算子，而且二進位檔很小。
- 支援多元平台：包含 Android 和 iOS 裝置、嵌入式 Linux 及微控制器，並善用平台 API 來加速推論。
- 提供多種語言的 API：包含 Java、Swift、Objective-C、C++ 和 Python。
- 高效能：在支援的裝置上執行硬體加速、提供針對裝置進行最佳化的核心，以及預先融合的啟用和偏誤。
- 模型最佳化工具：包含量化功能，可在不犧牲準確率的情況下，縮減模型的大小並提升效能。
- 有效率的模型格式：使用已針對小型檔案和可攜性進行最佳化的 FlatBuffer。
- 預先訓練模型：適用於常見的機器學習工作，可針對應用程式進行自訂。
- 範例和教學課程：說明如何在支援的平台上部署機器學習模型。

## 開發工作流程

TensorFlow Lite 的使用工作流程包含下列步驟：

1. **選擇模型**

   使用自己的 TensorFlow 模型、在線上尋找模型，或是從我們的預先訓練模型中進行挑選，並選擇直接套用或是重新訓練。

# TensorFlow Lite 範例應用程式

## TensorFlow Lite 範例應用程式

收錄多種 TensorFlow Lite 應用程式。

### 圖片分類

使用預先訓練模型，來測試圖片分類解決方案。該模型可辨識出行動裝置相機輸入畫面中 1000 種不同類型的物件。

在 Android 上試用
在 iOS 上試用
在 Raspberry Pi 上試用

### 物件偵測

瞭解這款應用程式如何運用預先訓練模型，為行動裝置相機輸入畫面中可辨識的不同物件 (約 1000 種)，繪製定界框並加上標籤。

在 Android 上試用
在 iOS 上試用
在 Raspberry Pi 上試用

### 姿勢估測

瞭解這款應用程式如何估測影像中人物的姿勢，

在 Android 上試用
在 iOS 上試用

### 語音辨識

瞭解這款應用程式如何透過麥克風辨別關鍵字，並傳回所說字詞的機率分數。

### 手勢辨識

使用 TensorFlow.js，訓練類神經網路辨識你的網路攝影機所捕捉到的手勢，然後使用 TensorFlow Lite 轉換模型，以便在你的裝置上執行推論。

### 智慧回覆

產生回覆建議，以輸入對話式即時通訊訊息。

# TensorFlow Lite 支援模型

## 最佳化的模型，適用於一般在行動裝置及邊緣上的使用案例

將最先進且經過最佳化的研究模型，輕鬆部署到行動裝置及邊緣裝置上。

### 圖片分類

辨識數百個物件，包含人物、活動、動物、植物和地點。

查看模型 →

### 物件偵測

使用定界框來偵測多個物件。沒錯，這也能偵測貓和狗。

查看模型 →

### 姿勢估測

估測單人或多人的身體姿勢。想像種種可能性，包括火柴人舞蹈派對。

查看模型 →

### 智慧回覆

產生回覆建議，以輸入對話式即時通訊訊息。

查看模型 →

### 區隔

透過極度準確的定位功能和語意標籤，精確指出物件外型。訓練內容包含人類、地點和動物等等。

查看模型 →

### 風格轉換

為輸入圖片套用任何樣式，建立新的藝術圖片。

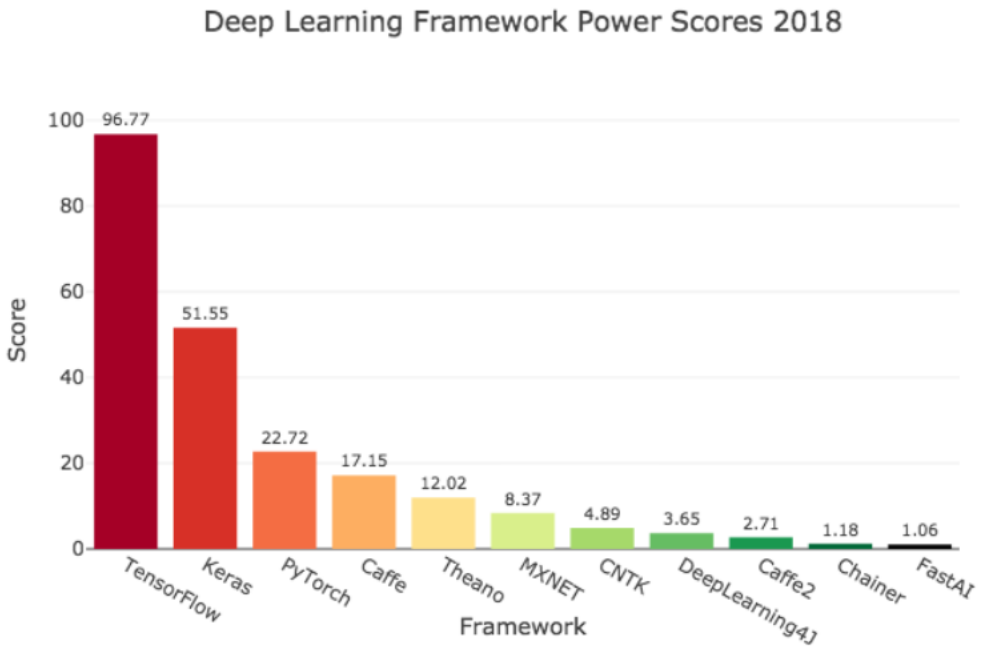
查看模型 →

# Case Study: Pixelopolis

- 使用TF-lite與手機相機功能實現的自動駕駛車
- https://blog.tensorflow.org/2020/07/pixelopolis-self-driving-car-demo-tensorflow-lite.html?hl=zh-tw

# Tensor Flow 2.0

- 現今重要且強大的深度學習框架之一
- Developed by Google and open source
- Alphago和Google Cloud Vision建立在TensorFlow之上




**Keras + TensorFlow =**
**更容易建構神經網路！**

Sources: https://www.tensorflow.org/?hl=zh-tw
https://medium.com/@kozyr_91350/chinese-all-about-tensorflow-f1e2ab1b89b1
https://ithelp.ithome.com.tw/articles/10215969

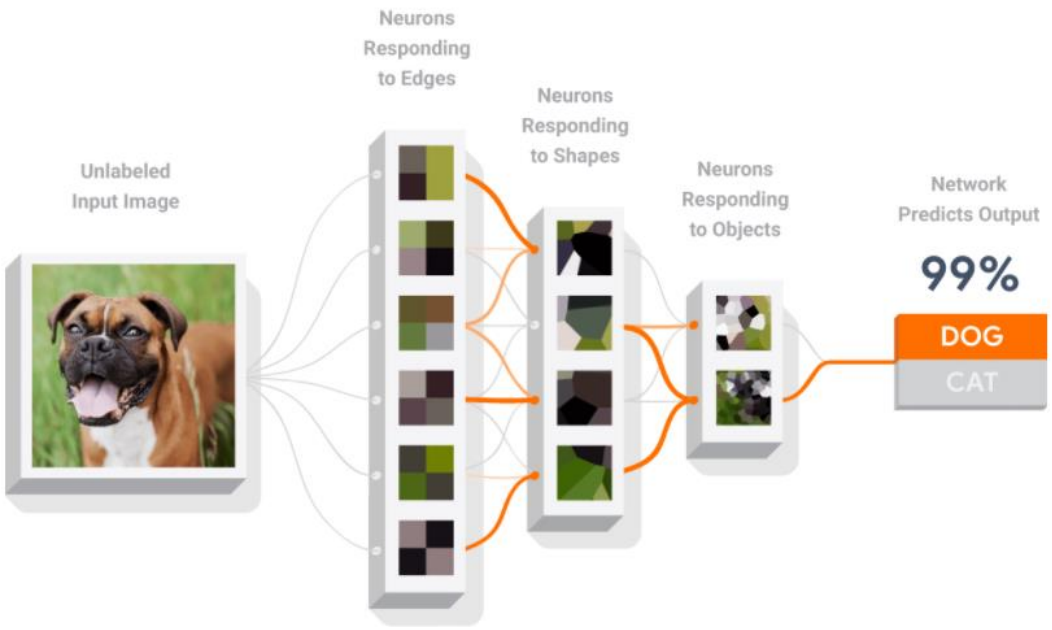
# Tensor (張量)與Flow

- 多維數據的容器

機器學習簡介 | 解決機器學習問題的步驟 | 類神經網路的解剖學 | 訓練類神經網路

## 類神經網路的解剖學

類神經網路是一種模型，經過訓練後可辨識模式。這種模型是由多個輸入層和輸出層所組成，且包含至少一個隱藏層。每一層中的神經元都會學習日益抽象的資料呈現。例如，在這個視覺圖表中，我們可以看到神經元是如何偵測到線條、形狀和紋理的。有了這些資料呈現 (也就是神經元學習到的特徵)，就能將資料分類。

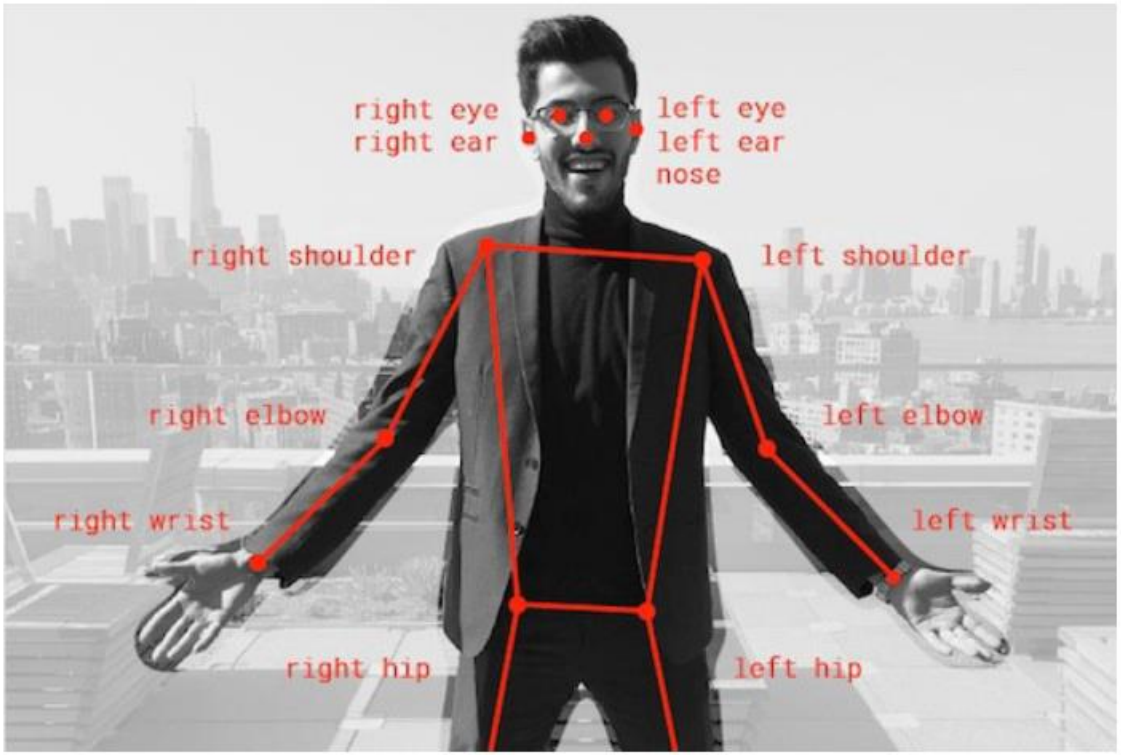




Source: https://kknews.cc/zh-tw/news/ma2g9n9.html

# 使用者可在瀏覽器內訓練並執行模型

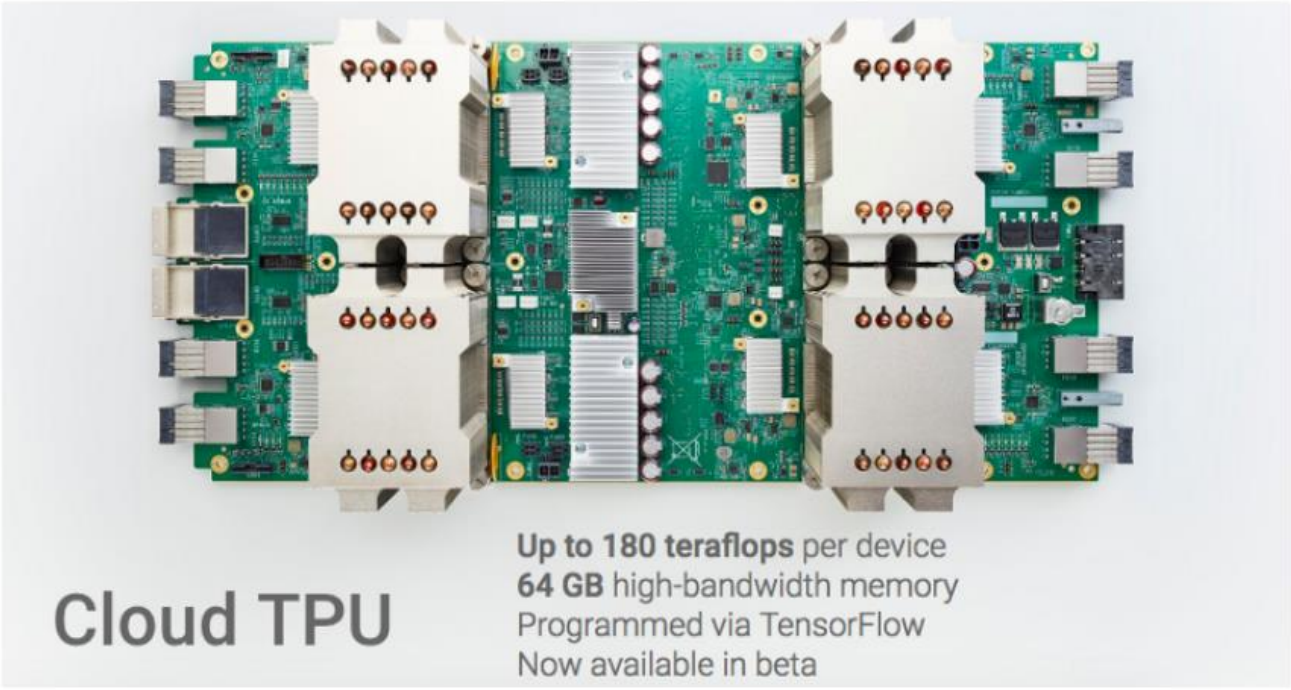
說到JavaScript，您可運用TensorFlow.js，在瀏覽器內訓練並執行模型。好好地研究這段酷炫的demo吧！當您回來時，我仍在這裡等您喔！



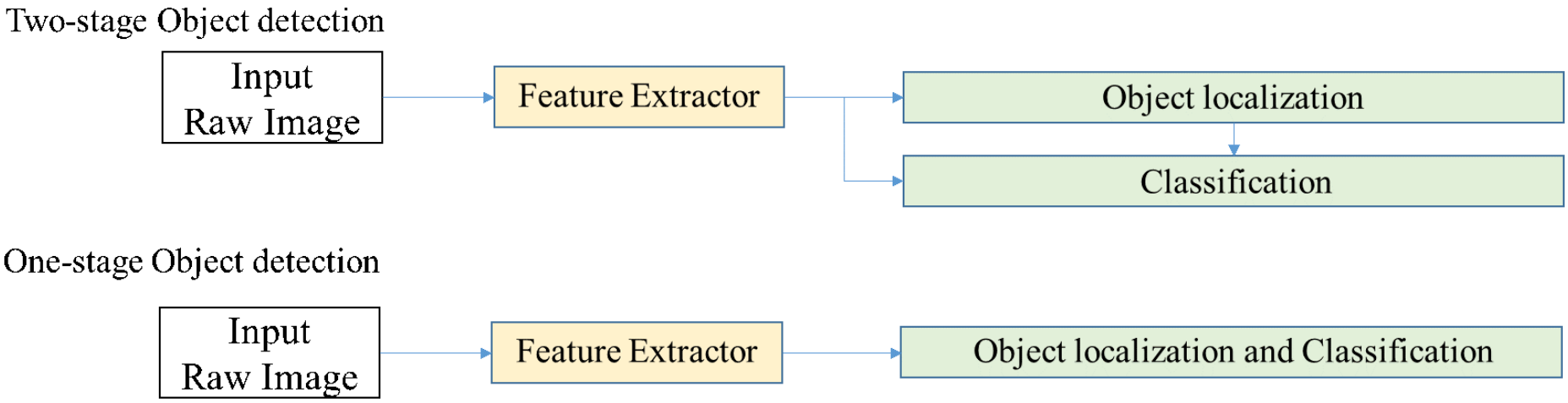

運用TensorFlow.js，在瀏覽器內進行即時人體姿勢評斷。請開啟您電腦上的攝影機進行一段demo，或著不要離開您的椅子，¯_(ツ)_/¯，由您決定。

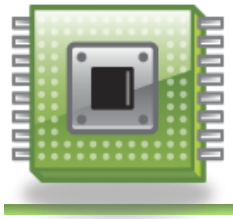
# 特製的硬體效能更佳

如果您已厭倦等候CPU將資料處理完畢，以訓練您的神經網路，現在，您可以取得搭載了Cloud TPU，專門設計用來處理大量資料的硬體。TPU的T，意指「張量（Tensor）」，跟TensorFlow的Tensor有一樣的意思。巧合嗎？我認為不是！幾週前，Google宣佈第三版的TPU，目前發展到預覽版本。



# Single Shot MultiBox Detector (1/3)



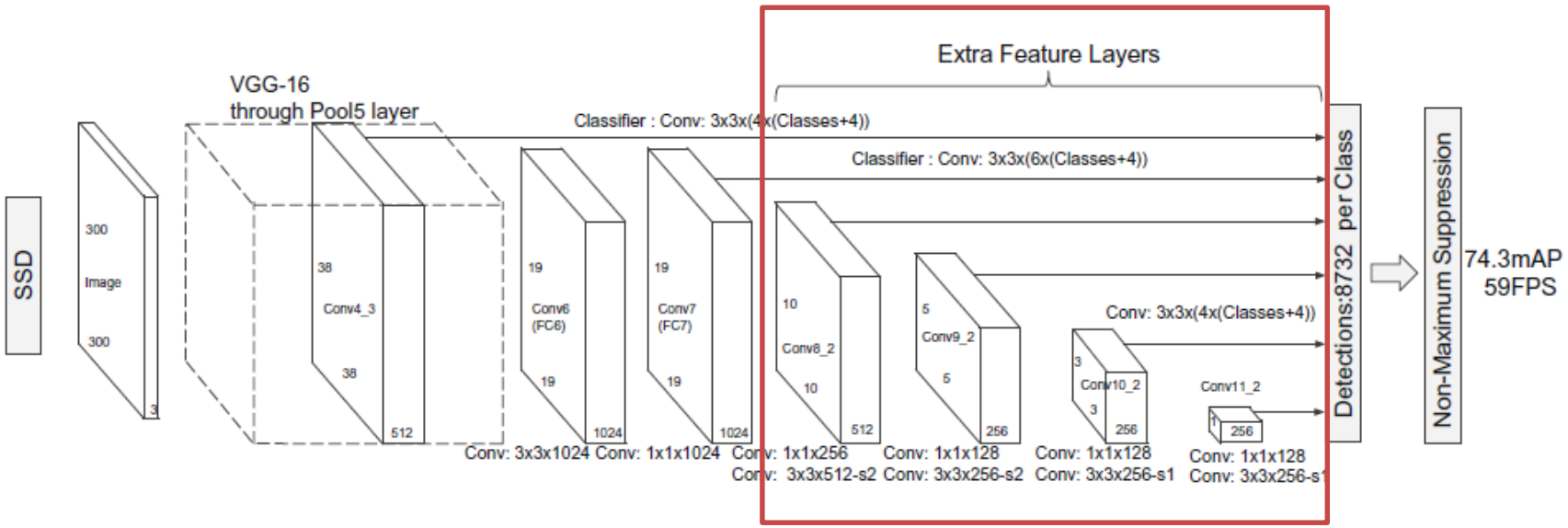


# Single Shot MultiBox Detector (2/3)

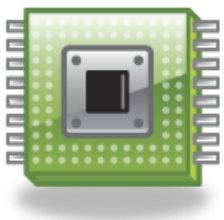
- **Faster than YOLO(v1) and has comparable accuracy of Faster R-CNN**
- **Much better accuracy with small input image size compare to other single stage methods**
- **Features**
  - **Multi-scale feature maps for detection**
  - **Convolutional predictors for detection**
  - **Default boxes and aspect ratios**

[2] W. Liu et al. , “SSD: Single Shot MultiBox Detector”, *2016 European Conference on Computer Vision(ECCV)*, pp. 21-37, 2016.

# Single Shot MultiBox Detector (3/3)




**Backbone (feature extractor): VGG16**



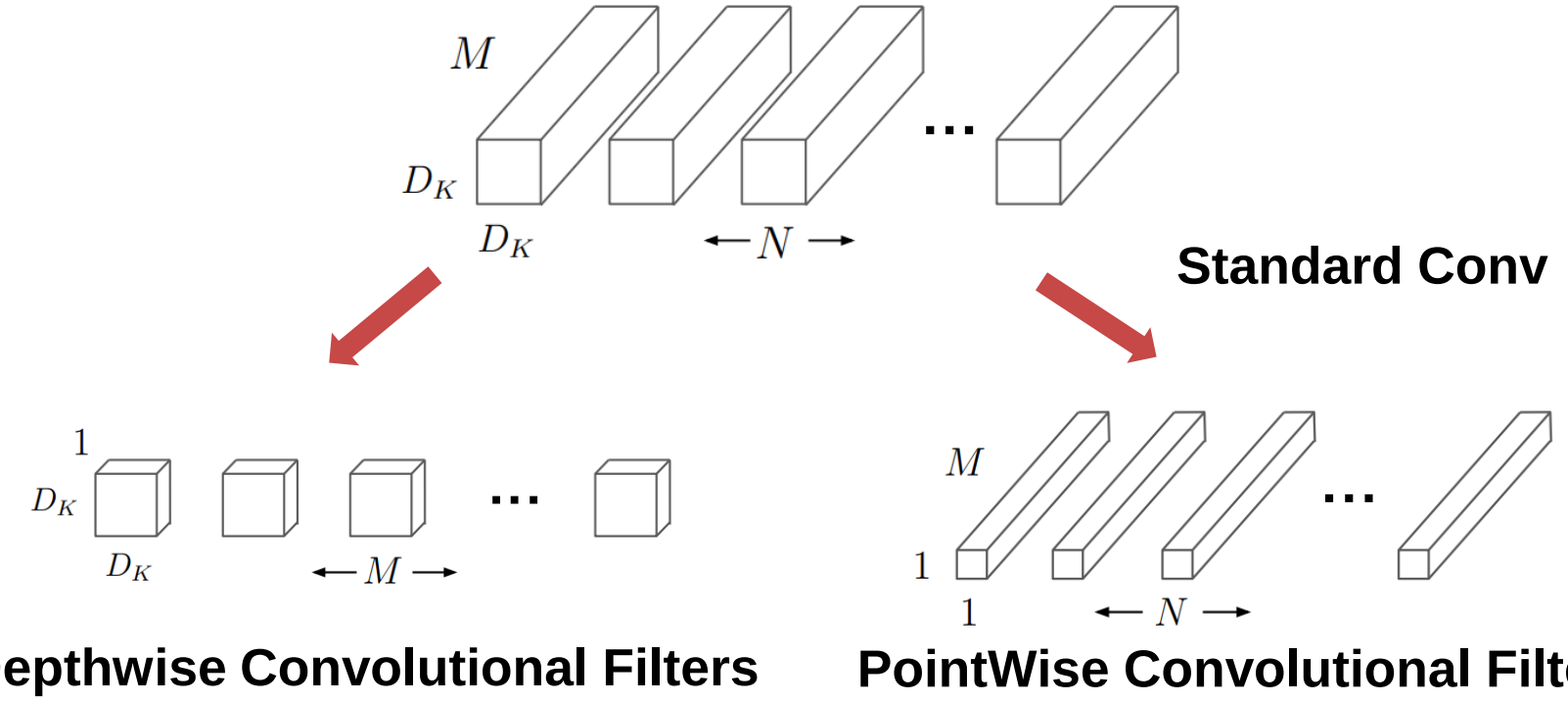
# Building Small and Efficient Networks

- **Compress pretrained networks**
- **Train small networks**
- **Develop network architectures that allows a model developer to specifically choose a small network that matches the resource restrictions (latency, size) for their application**

[3] A. G. Howard et al. , “MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications”, *Google Inc. 2017*

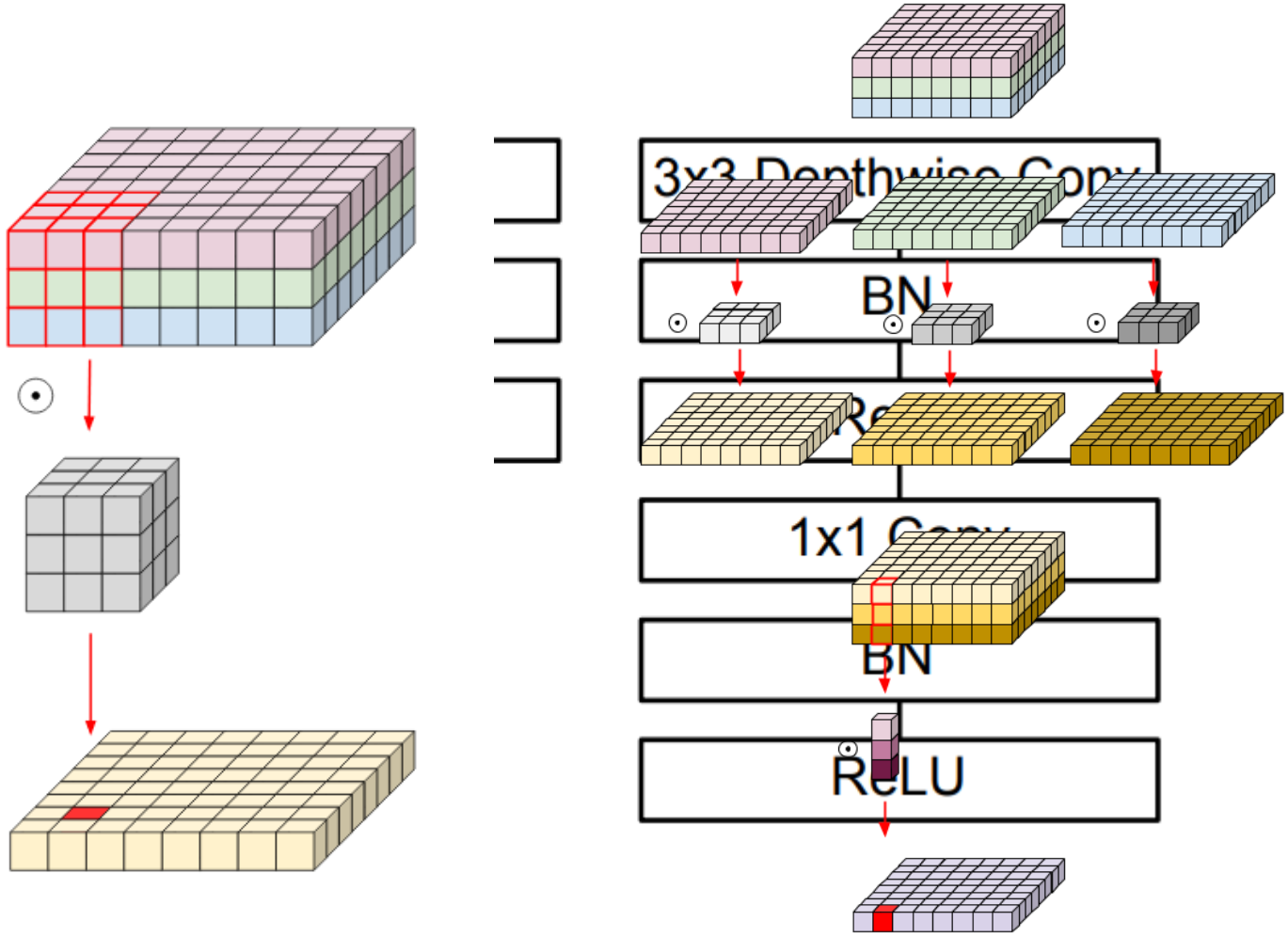
# Mobilenet (1/2)

- **Less Calculation**
- **Similar performance**




[3] A. G. Howard et al. , “MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications”, *Google Inc. 2017*

# Mobilenet (2/2)




[4] https://medium.com/@zurister/depth-wise-convolution-and-depth-wise-separable-convolution-37346565d4ec

# 模型比較

Table 1: Average precision at IoU 0.95 and 0.50.

| # | Model | Framework | AP [IoU=0.95] | AP [IoU=0.50] |
|---|---|---|---|---|
| 1 | Faster RCNN (ResNet-101) | Tensorflow | **0.245** | **0.476** |
| 2 | YOLOv3-416 | Darknet | 0.143 | 0.367 |
| 3 | Faster RCNN (Inception ResNet-v2) | Tensorflow | **0.317** | **0.557** |
| 4 | YOLOv2-608 | Darknet | 0.198 | **0.463** |
| 5 | Tiny YOLO-416 | Darknet | 0.035 | 0.116 |
| 6 | SSD (Mobilenet v1) | Tensorflow | 0.094 | 0.233 |
| 7 | SSD (VGG-300) | Tensorflow | 0.148 | 0.307 |
| 8 | SSD (VGG-500) | Tensorflow | 0.183 | 0.403 |
| 9 | R-FCN (ResNet-101) | Tensorflow | **0.246** | **0.486** |
| 10 | Tiny YOLO-608 | Darknet | 0.06 | 0.185 |
| 11 | SSD (Inception ResNet-v2) | Tensorflow | 0.116 | 0.267 |
| 12 | SqueezeDet | Tensorflow | 0.003 | 0.012 |
| 13 | R-FCN | Tensorflow | 0.124 | 0.319 |

Kim, Chloe & Maktabdar Oghaz, Mahdi & Fajtl, Jiri & Argyriou, Vasileios & Remagnino, Paolo. (2018). A Comparison of Embedded Deep Learning Methods for Person Detection.

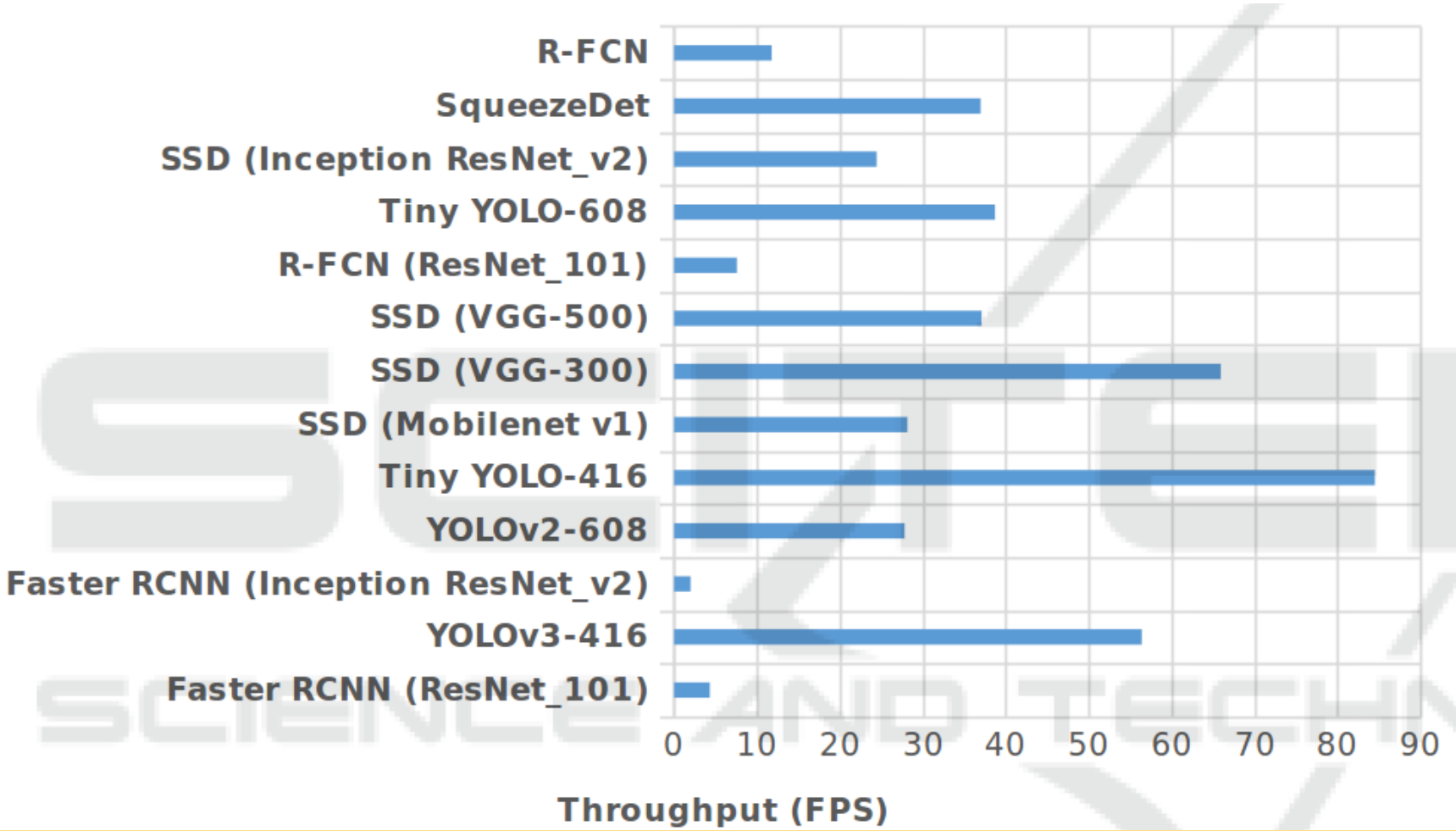
# 模型比較

Table 2: Total latency of inference in both CPU and GPU modes.

| # | Model | CPU Latency (S) | GPULatency (S) |
|---|---|---|---|
| 1 | Faster RCNN (ResNet-101) | 3.271 | 0.232 |
| 2 | YOLOv3-416 | 5.183 | 0.017 |
| 3 | Faster RCNN (Inception ResNet-v2) | 10.538 | 0.478 |
| 4 | YOLOv2-608 | 11.303 | 0.035 |
| 5 | Tiny YOLO-416 | 1.018 | 0.011 |
| 6 | SSD (Mobilenet v1) | **0.081** | 0.03 |
| 7 | SSD (VGG-300) | 0.361 | **0.015** |
| 8 | SSD (VGG-500) | 0.968 | **0.026** |
| 9 | R-FCN (ResNet-101) | 1.69 | 0.131 |
| 10 | Tiny YOLO-608 | 2.144 | **0.025** |
| 11 | SSD (Inception ResNet-v2) | **0.109** | 0.04 |
| 12 | SqueezeDet | **0.14** | **0.027** |
| 13 | R-FCN | 3.034 | 0.084 |

Kim, Chloe & Maktabdar Oghaz, Mahdi & Fajtl, Jiri & Argyriou, Vasileios & Remagnino, Paolo. (2018). A Comparison of Embedded Deep Learning Methods for Person Detection.

# 模型比較




Kim, Chloe & Maktabdar Oghaz, Mahdi & Fajtl, Jiri & Argyriou, Vasileios & Remagnino, Paolo. (2018). A Comparison of Embedded Deep Learning Methods for Person Detection.