基於深度學習輔助系統平台標註技術

- 優化 R 語言、使用 R 語言用途
- 導入 RCPP、python 優化提升速度
- 資料前處理與訓練過程
- Openslide 介紹
- 其他 docker、GPU、CPU、tensorflow、pytorch、執行續、平行化、批次處理資料
- 未來展望

優化 R 語言、使用 R 語言用途

使用 R 語言用途

(一) Python 的功能

Python 被定位成一種能被廣泛使用的程式語言，應用多、資源多、支援多，入門的門檻低，相對於其他語言更容易撰寫且直觀，我們可以把它想像成一個多功能的工具，可以用在網路爬蟲、資料整理、機器學習、深度學習、資料視覺化、自動排程與部署等。

(二) R 語言的功能

R 語言則更常被用在統計分析。R 的設計起源來自於「數學」，是一種適用於統計運算與畫圖的語言與環境。我們可以將其比喻成一種專攻某種功能的工具，適用於需要頻繁的數據檢驗時，能夠快速建模。而且 R 的視覺化功能非常強大，適合製作視覺化圖表與統計分析。

差異處	Python	R 語言
易學程度	簡單、直觀 學習門檻較低	學習門檻較較高 高級功能複雜性也較高
語言熱門度	Tiobe index for July 2021 排行第三	Tiobe index for July 2021 排行第十二
應用程度	應用範圍廣，如：機器學習	專研統計模型與數據分析
視覺化效果	較差	較佳，能製作出漂亮的圖形
資料 導入相容性	Python支持多種資料格式， 如Excel、CSV、JSON，甚至SQL表， 皆可透過相應套件導入Python	可從Excel、CSV和文本文件導入。 在Windows系統上， 處理中文變數編碼稍具挑戰性， 若在Linux系統上操作，會較為順暢。
業界應用範圍	Web應用系統製作與串接容易	Web應用系統製作與串接較具有挑戰性

醫學統計

醫學統計是指將統計方法應用在生物或醫學的研究資料，醫學統計包含研究設計，資料蒐集，資料處理與資料分析．每一組量化資料，可以有許多不同分析的統計方法，沒有絕對正確且單一統計分析方法，而是有一些合理的統計分析方法，可以在所合理的分析方法中，選擇相對適合的統計分析方法，但對於任一組量化資料，確是有些絕對錯誤的分析方法．

基本統計學常將統計分析資料分成 描述性統計與推論性統計，但這些類別之間並沒有明顯的界線．描述性統計（descriptive statistics）是以數字，圖表說明資料的特徵，將資料作最佳的呈現．而 推論性統計（inferential statistics）是從資料中對群體取得一般化的結論．另外，統計分析資料的結果，經常使用來預測與決策，預測（prediction），是從資料中預測各種事件可能發生的機會或數值．決策（decision making）則是依據統計資料，做出決策，決策 或 預測 常歸類成 推論性統計．預測與決策通常針對更大的群體，而不只限於資料樣本．

統計的描述性統計與推論性統計主要是分析資料的方法，研究者利用這些分析資料的方法，進行研究與回答研究的問題．在分析資料前，研究者必須仔細思考研究問題的本質是什麼？研究問題的本質，將會深深地對影響研究設計，描述與推論．例如，研究標靶藥物治療肺癌的結果，研究者必須思考研究問題的本質是（a）標靶藥物治療肺癌的結果會顯現在減少腫瘤體積？（b）顯現在延長沒有肺癌的時間？或（c）顯現在延長肺癌的存活時間？同一疾病但不同的研究問題，不同的病期或病程，則會有不同的研究設計，描述與推論．

優化 R 語言

	編譯器 Compiler	直譯器 Interpreter
編譯方式	一次性編譯	逐行編譯
運行速度	快	慢
輸出檔案	.exe 檔案	無
支持語言	C、C++	Python、Ruby、MATLAB

R 語言之所以慢，主要是因為它是一種執譯型語言、其資料操作通常基於整個物件、缺少原生的多執行緒支援、且某些操作都依賴函數調度機制。然而，透過改進程式碼的向量化、使用效率速度較好 package、可以使用 C/C++或 python 撰寫程式、使用平行化運算少寫迴圈，有可能顯著提高 R 語言的運作效率。其中，程式碼的向量化尤其關鍵。

1. 向量化例如:R 向量預算 C()、list()、which 函數等等、矩陣 matrix 與陣列 array
2. 使用效率速度較好 package: 例如，data.table 和 dplyr 套件對於資料操作提供了超越基礎 R 功能的效能，使用 apply()、*apply()函數取代 for 迴圈
3. Rcpp 是一個可以讓 R 透過 C++ 語言提升執行效能的套件，可使用 cppFunction()嵌入 C++語言或 sourceCpp("cpp_sum.cpp")呼叫已寫好的 C++程式碼
4. Python reticulate::py_run_string()reticulate 套件
5. 資料平行處理，使用支援平行處理套件或函數，當資料量大時速度較為顯著改善，多執行續、多線呈提高 CPU 使用率同時也號記憶體，另外垃圾回收機制釋放記憶體
6. 使用殂存資料較小讀取較快的檔案類型二進制檔案存取

https://medium.com/ching-i/%E5%A4%9A%E5%9F%B7%E8%A1%8C%E7%B7%92-de16f92944c8

- future 在 R 語言中提供統一的平行和分散式處理框架
- future.apply 可以取代 base R 提供的 apply 族函數
- future.batchtools 使用 batchtools 實現並行和分散式處理
- batchtools Map 函數的平行實現，用於高效能運算系統和分散式處理，可以單機多核心並行也可以多機並行，還提供了一種抽象的機制去定義大規模電腦實驗。
- multidplyr 是 dplyr 的後端，多核心環境實現資料分塊，提高平行處理效能
- disk.frame 是基於磁碟的超出記憶體容量的快速並行資料操作框架
- parallelMap R package to interface some popular parallelization back-ends with a unified interface
- big.data.table 基於 data.table 的分散式平行計算

Openslide 介紹

OpenSlide 是一個 C 函式庫，它提供了一個簡單的介面來讀取整個幻燈片影像，也稱為虛擬玻片，這是數位病理學中使用的高解析度影像。這些圖像在未壓縮時可能會佔用數 10 GB，因此無法使用標準工具或庫輕鬆讀取，而這些工具或庫是為可以輕鬆解壓縮到 RAM 中的圖像而設計的。整個幻燈片影像通常是多重解析度的； OpenSlide 允許以最接近所需縮放等級的解析度讀取少量影像資料。

OpenSlide can read virtual slides in several formats:

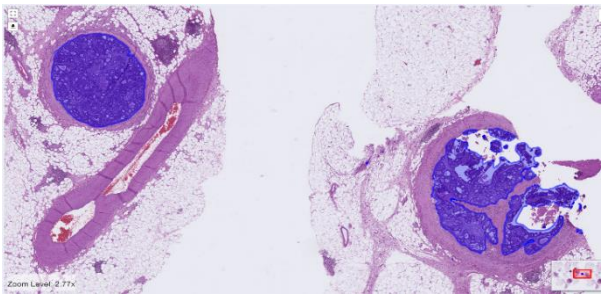
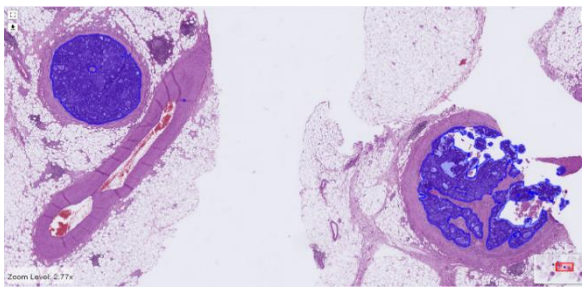
- Aperio (`.svs`, `.tif`)
- DICOM (`.dcm`)
- Hamamatsu (`.ndpi`, `.vms`, `.vmu`)
- Leica (`.scn`)
- MIRAX (`.mrxs`)
- Philips (`.tiff`)
- Sakura (`.svslide`)
- Trestle (`.tif`)
- Ventana (`.bif`, `.tif`)
- Generic tiled TIFF (`.tif`)

https://openslide.org/api/python/

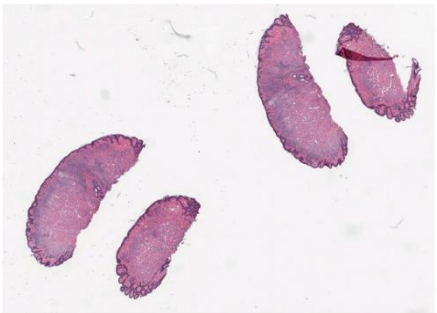
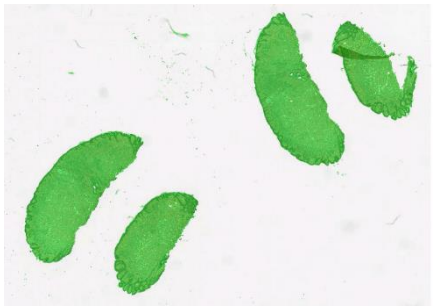
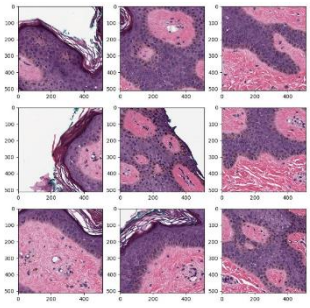
資料前處理與訓練過程

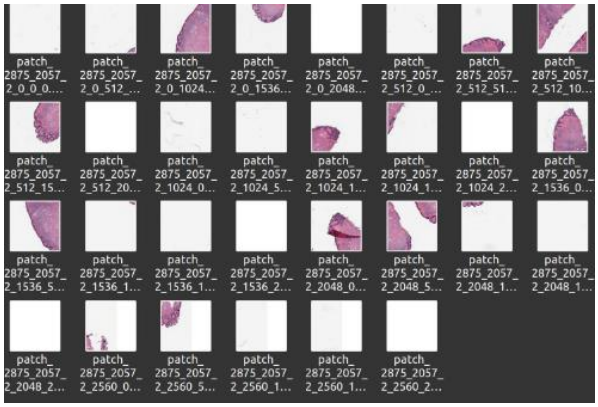
1. 資料錢處理:

蒐集玻片並標註，標註過程形狀為多邊形

在使用 openslide 進行影像分割

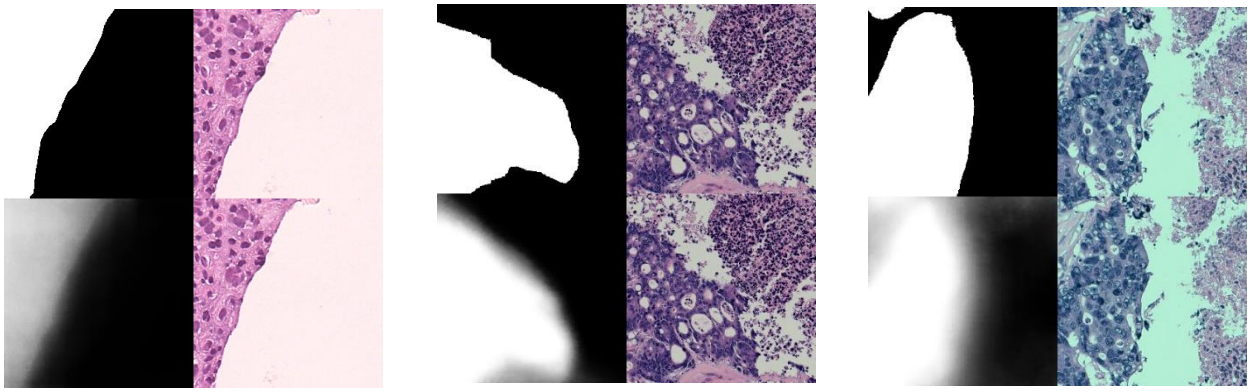






2. 影像分割與形態學影像處理與萃取特徵

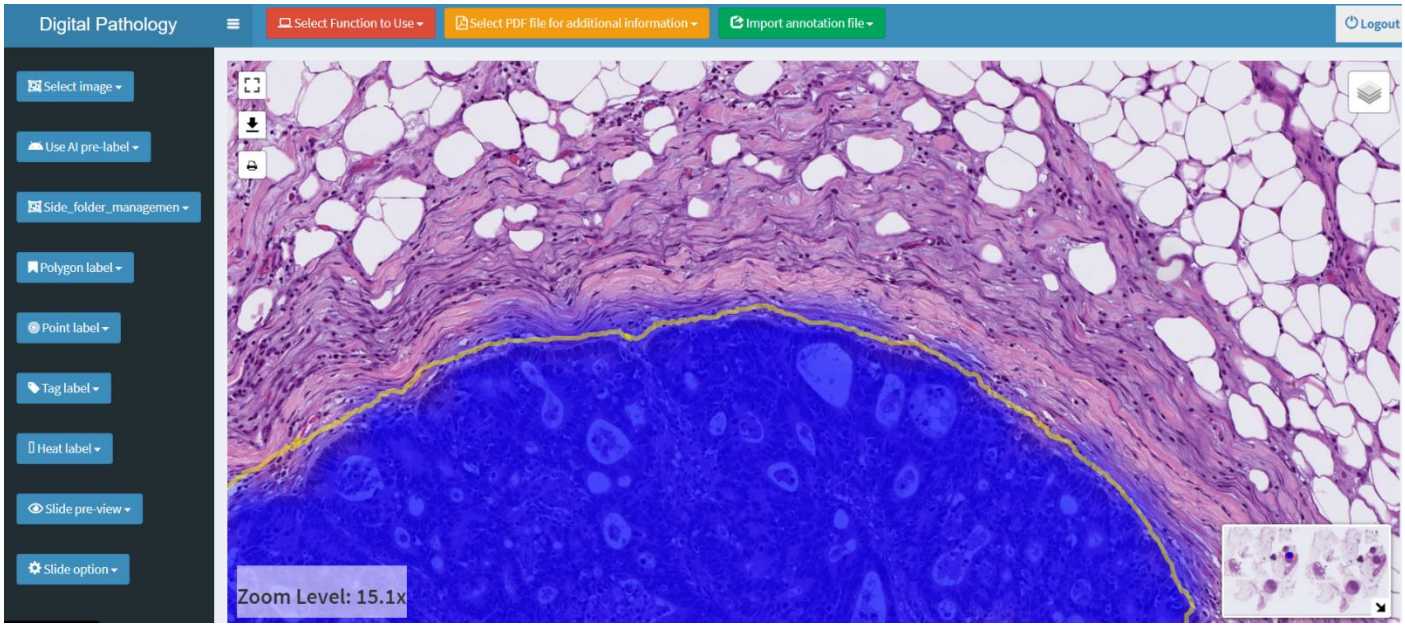
影像分割

影像二值化被多邊形標註內的設值為 1 沒被標記到的設值為 0 也就是黑色

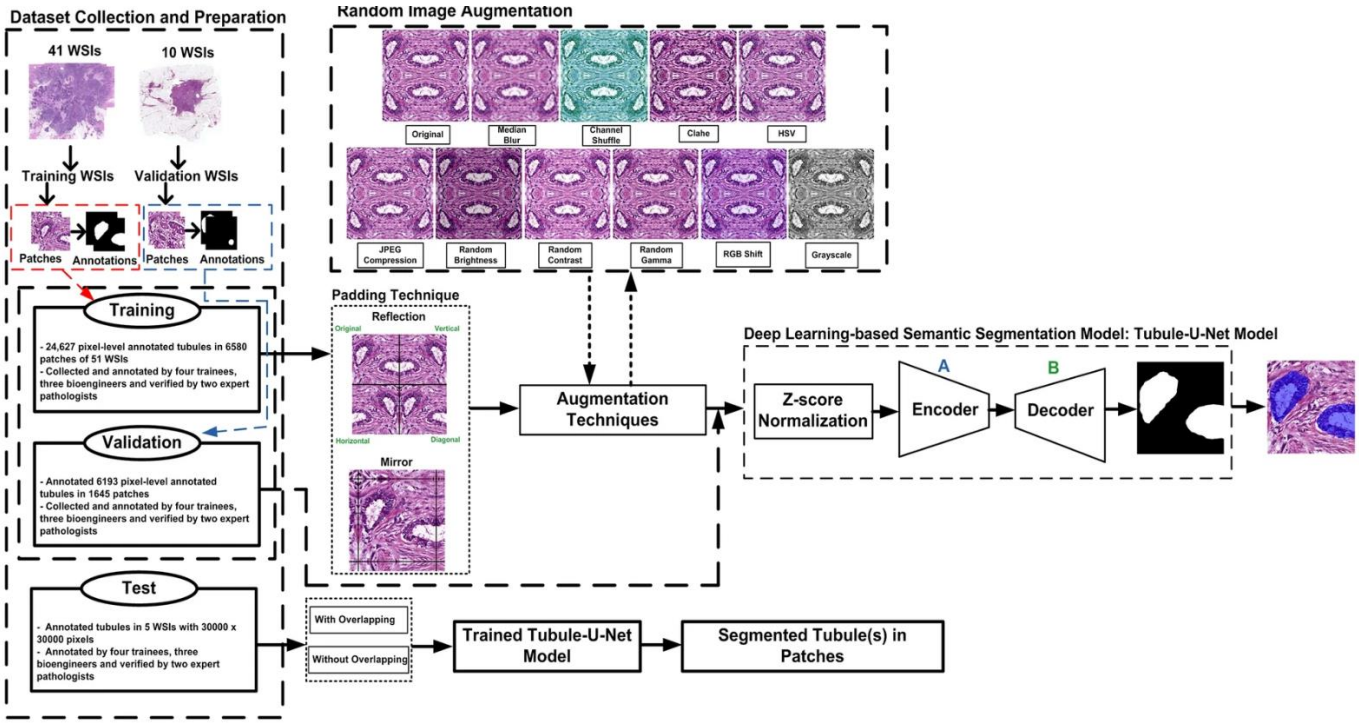


形態學影像處理

侵蝕與膨脹 Erosion and Dilation、多邊形截彎取直或者簡化多邊形 polysimplify()



訓練



- 其他 docker、GPU、CPU、tensorflow、pytorch、執行續、平行化、批次處理資料

GPU、CPU 搭配，GPU 在訓練時特別在圖像訓練時，CPU 特別在城市的執行續，多線呈，大量數據讀取並進行複雜運算，

CPU 對於現代運算任務有幾個明顯的優點：

· 靈活性－CPU 是一種通用處理器，可以處理許多任務，以及多個活動之間的多任務。

· 在許多情況下更快——在處理 RAM 中的資料處理、I/O 操作和作業系統管理等操作時，CPU 比 GPU 更快。

· 精度－CPU 可以支援比 GPU 精度更高的中等數學運算，這對於許多用例來說都很重要。

· 高速緩存－CPU 有一個很大的本地高速緩存，這讓它們可以處理大量的線性指令。

· 硬體相容性－CPU 相容於所有類型的主機板和系統設計，而 GPU 需要專門的硬體支援。

GPU 的獨特優勢包括：

· 高資料吞吐量－GPU 可以對許多資料點並行執行相同的操作，因此它可以以 CPU 無法比擬的速度處理大量資料。

· 大規模並行－一個 GPU 有數百個核心，使其能夠執行大規模平行計算，例如矩陣乘法。

· 適用於專門的用例－GPU 可以為深度學習、大數據分析、基因組定序等專門任務提供巨大的加速。

損失函數的使用

DICE、Focal、Jaccard_IoU、MSE、Tversky

優化函數

Adadelta、Adagrad、Adam、Adamax、Ftrl、Nadam、RMSprop、SGD

- **docker**

能夠快速地進行開發、交付、部署、測試應用程式。透過 Docker 將當前的環境與應用程式封裝在一起，可以更方便地移植、部署到任何一個有安裝 Docker 的平台上。

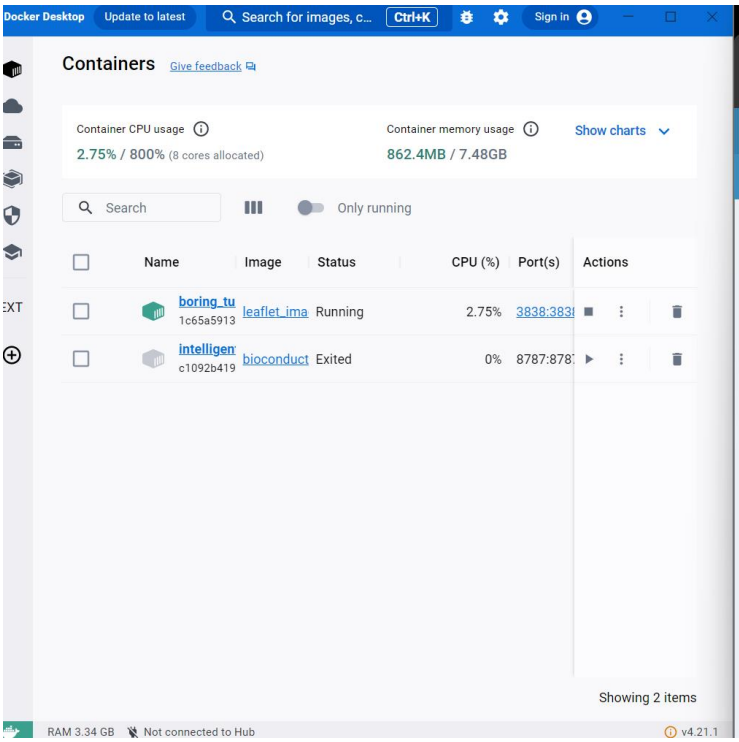
✓ 更快速的交付與部署

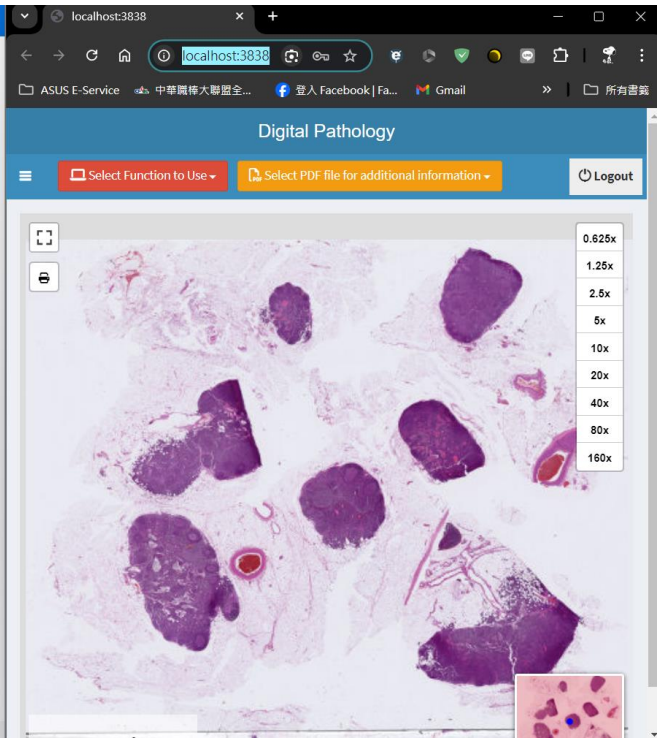
✓ 高效能的虛擬化

✓ 容易移植與擴展

✓ 管理簡單化

✓ 佔用記憶體空間較小





3. 未來展望

分散式處理資料

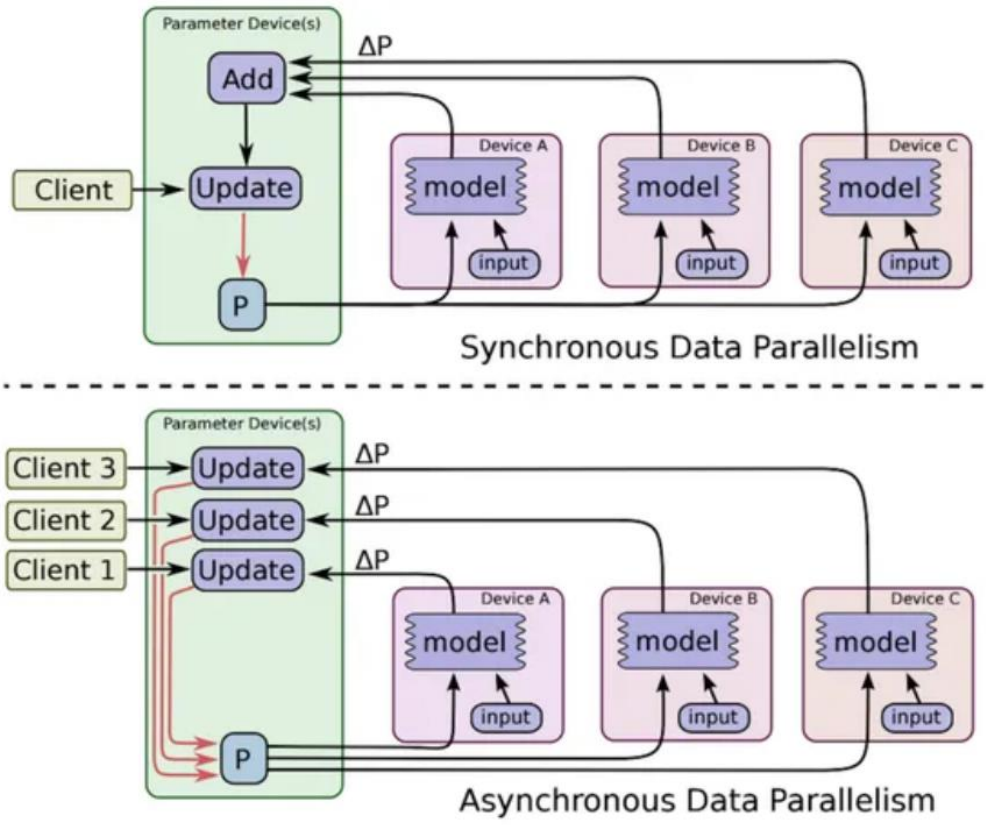
分散式訓練就是指將模型放置在很多台機器且每台機器有多個 GPU 上進行訓練，之所以使用分散式訓練的原因有兩種：第一、模型在一塊 GPU 上放不下，第二、使用多塊 GPU 進行平行計算能夠加速訓練。但需要注意的是隨著使用的 GPU 數量增加，各個設備之間的通訊會越複雜，導致訓練速度下降。

分散式訓練主要分為兩種類型：資料平行化（Data Parallel）、模型平行化（Model Parallel）。

✓ 資料平行化（Data Parallel）

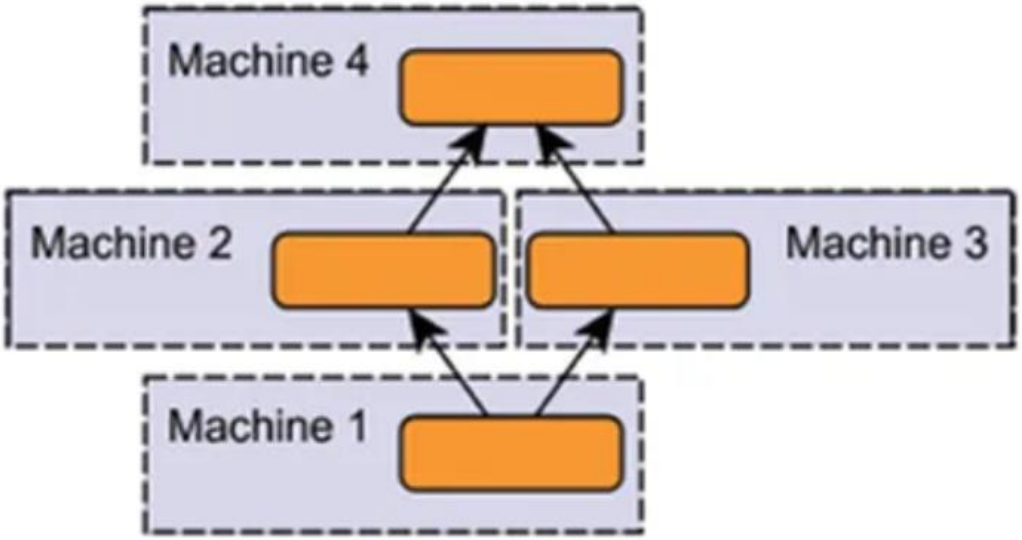
當數據量非常大，並且模型架構能夠放置在單個 GPU 上時，就可以採用資料平行化的方式來進行分工合作。
做法是依照一些規則將數據分配到不同的 GPU 上，並且每個 GPU 都有相同的模型架構，也就是會在每個 GPU 上複製一份相同的模型，各自進行訓練後，將計算結果合併，再進行參數更新。
參數更新的方式又分為同步及異步:

同步（synchronous）: 所有的 GPU 在訓練時會等待其他 GPU 計算完畢後，才會進行一次參數更新，因此訓練速度上會比使用異步的方式來得慢。但因為在更新參數時會合併其他計算結果，相當於增加了 batch size 的大小，對於訓練結果有一定的提升。
非同步、異步（asynchronous）: 每個 GPU 各自進行訓練和參數更新，不須等待其他 GPU 計算完畢。能夠提升訓練速度，但可能會產生 Slow and Stale Gradients（梯度失效、梯度過期）問題，收斂過程較不穩定，訓練結果會比使用同步的方式差。



✓ 模型平行化（Model Parallel）
當模型架構太大以至於在一個 GPU 放不下時，可以採用模型平行化的方式來將模型拆解並分配到不同的 GPU 上。

由於模型層與層之間通常有依賴性，也就是指在進行前向傳播、反向傳播時，前面及後面的層會作為彼此的輸入和輸出，在訓練速度上會有一定的限制，因此若非使用較大的模型較不建議採用模型平行化。若想提升訓練速度，可以選擇較容易進行平行運算的 module，ex: Inception。



- ✓ https://medium.com/ching-i/pytorch-%E5%88%86%E6%95%A3%E5%BC%8F%E8%A8%93%E7%B7%B4-distributeddataparallel-%E6%A6%82%E5%BF%B5%E7%AF%87-8378e0ead77
- ✓ https://data-flair.training/blogs/distributed-tensorflow/

4. Numba、cupy

✓ **CuPy**

CuPy 是一個 Python 函式庫，與 NumPy 和 SciPy 陣列相容，為 G PU 加速運算而設計。將 NumPy 換成 CuPy 語法，您可以在英偉達 CUDA 或 AMD ROCm 平台上執行程式碼。這讓您可以使用 GPU 加速執行與陣列相關的任務，從而更快地處理更龐大的陣列。

只要換掉幾行程式碼，就可以利用 G PU 的大規模並行處理能力來大幅加快索引、規範化和矩陣乘法等陣列運算。

CuPy 也支援存取低階 CUDA 功能。它允許使用 RawKernels 將 ndarray 傳遞給現有的 CUDA C/ C++程序，借助 Streams 簡化效能，並允許直接呼叫 CUDA Runtime API。

https://www.51cto.com/article/772497.html

✓ **Numba**

為了提高執行速度，Numba 會在執行前立即將 Python 位元組代碼轉換為機器碼。

Numba 可用於使用可呼叫的 Python 物件（稱為修飾器）來最佳化 CPU 和

GPU 功能。修飾器是一個函數，它將另一個函數作為輸入，進行修改，並將修改後的函數傳回給使用者。這種模組化可減少編程時間，並提高 Python 的可擴展性。

Numba 也可與 NumPy 結合使用，後者是一個複雜數學運算的開源 Python 庫，專為處理統計數據而設計。呼叫修飾器時，Numa 將 Python 和/或 NumPy 程式碼的子集轉換為針對環境自動最佳化的字節碼。它使用 LLVM，這是一個以 API 為導向的開源程式庫，用於以程式設計方式建立機器原生程式碼。 Numba 針對各種 CPU 和 GPU 配置，提供了多種快速並行化 Python 程式碼的選項，有時只需一條指令即可。與 NumPy 結合使用時，Numba 會為不同的陣列資料類型和佈局產生專用程式碼，進而優化效能。

- ✓ https://www.nvidia.cn/glossary/data-science/numba/

5. Tensorflow、pytorch

PyTorch 是一個開源的深度學習框架，建立於 Torch 之上，底層為 C++，並標榜 Python First，強調其為 Python 語言量身打造的，使用上就與 Python 專案的撰寫並沒有太大的差異，也能夠與 Python 的套件相整合。作為新手入門的選項，其優勢就是概念架構直觀、語法簡潔，輕量架構也讓模型能夠快速訓練。

TensorFlow 自從開放原始碼後，就成為創建深度學習模型時使用的熱門框架之一。TensorFlow 之所以席捲全球，除了免費的緣故之外，也因為他相對容易上手的特點，即便是機器學習的初學者，也能夠透過函式庫中的資料避免從零開始建構。

PyTorch 與 TensorFlow 有什麼差別？

同樣作為適合新手入門人工智慧領域的 PyTorch 與 TensorFlow，他們之間又有哪些差異呢？兩派各有其支持者，相對來說 PyTorch 更容易上手、框架靈活，有 Python 背景的情況能更輕鬆的使用。而 TensorFlow 則是勝在有完整的文章框架、模型與教程，模組都被封裝得相當精緻，並對程式碼進行了有效的縮減。

PyTorch 與 TensorFlow 的差異不小，PyTorch 語法較為簡潔之外，主打的動態圖設計也方便研究者調整及試驗。TensorFlow 則在計算效率上有優勢，而且由於開發的早，很多應用都是以 TensorFlow 為主，這對產業來說，定是以實際應用為主，即便後期新框架層出不窮，但除非有極大的改變，否則難以撼動 TensorFlow 在產業端的應用。

6. Kubernetes

Kubernetes 是一個協助我們自動化部署、擴張以及管理容器應用程式(containerized applications)的系統。相較於需要手動部署每個容器化應用程式(containers)到每台機器上，Kubernetes 可以幫我們做到以下幾件事情：

- ✓ 同時部署多個 containers 到一台機器上，甚至多台機器。

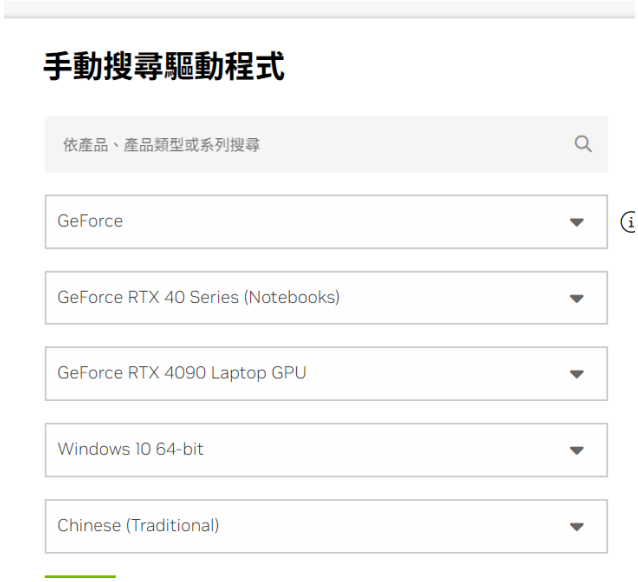
- ✓ 管理各個 container 的狀態。如果提供某個服務的 container 不小心 crash 了，Kubernetes 會偵測到並重啟這個 container，確保持續提供服務
- ✓ 將一台機器上所有的 containers 轉移到另外一台機器上。
- ✓ 提供機器高度擴張性。Kubernetes cluster 可以從一台機器，延展到多台機器共同運行。

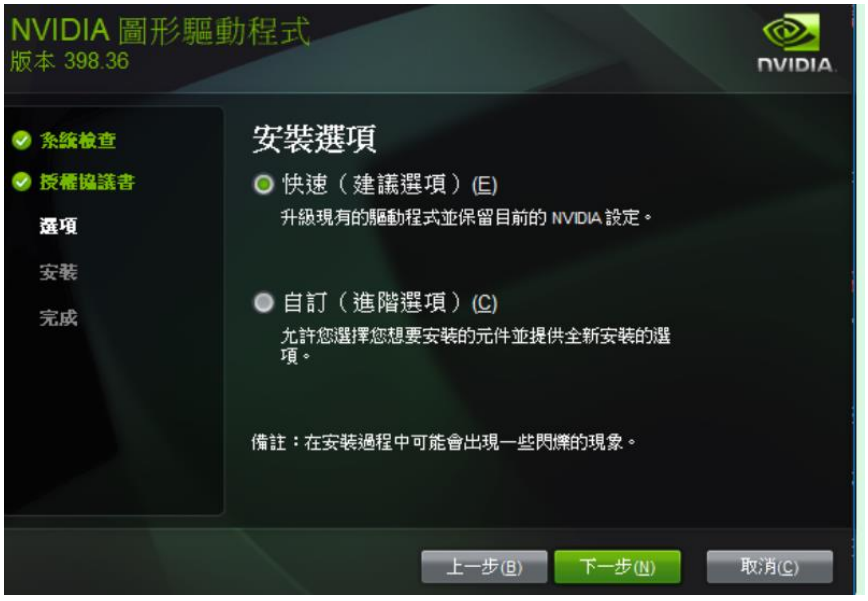
7. 環境設置

系統開發環境上，主程式以 R 語言去撰寫，程式環境版本為 R 4.1.3，R 語言速度優化使用 python 與 C++程式進行優化，硬體部分 NVIDIA GeForce RTX 3080Ti，128GB RAM，作業系統環境：Windows 10 專業版 64 位元。

下載 PyTorch 或 TensorFlow

下載 CUDA 驅動程式





下載 CUDA Toolkit

介紹 CUDA

NVIDIA 發明了 CUDA 程式設計模型並解決這些挑戰。CUDA 是用於圖形處理單元（graphical processing units，GPU）的平行運算平台和程式設計模型。透過 CUDA，您可以利用 GPU 的運算能力加速應用程式。

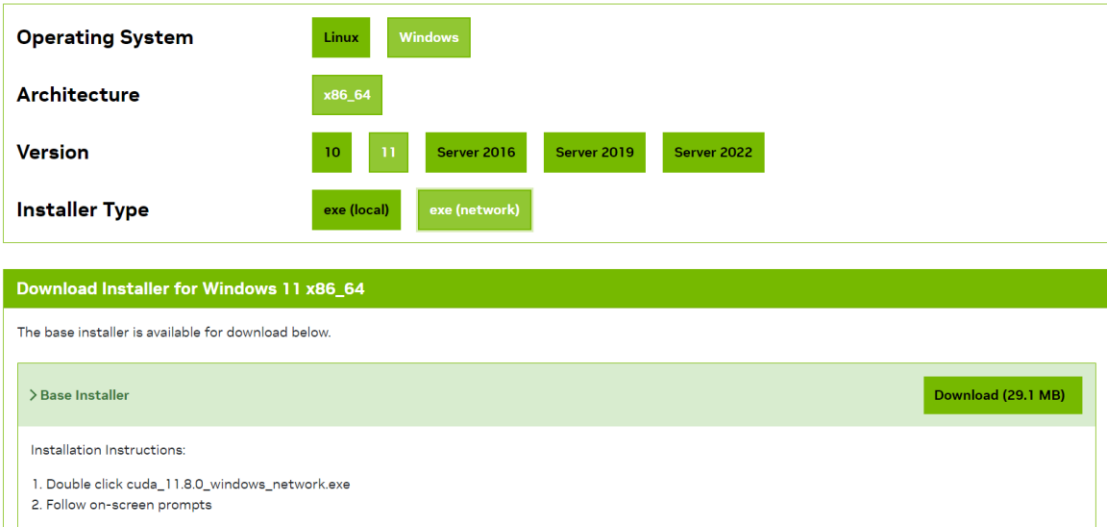
NVIDIA 於 2006 年 11 月推出 CUDA 的第一版，其軟體環境讓使用者使用 C 做為高階程式設計語言。透過 CUDA 加快數以千計的應用程式，其中包括推動機器學習和深度學習的函式庫和框架。

易於設計

為了方便採用，CUDA 提供了基於 C/C++ 的介面。CUDA 程式設計模型的一大好處是可讓您編寫純量程式。CUDA 編譯器使用程式設計中的語意抽象化，以此來利用 CUDA 程式設計的平行性。這可減輕程式設計的負擔。以下是關於 CUDA 程式設計模型的一些基礎知識。

CUDA 程式設計模型為程式設計師提供了三個主要程式語言延伸：

- ✓ CUDA 區塊（block） – 執行緒的集合或群組。
- ✓ 共用記憶體 – 所有執行緒（thread）在區塊中共用的記憶體。
- ✓ 同步屏障 – 讓多個執行緒能等到所有執行緒都到達特定執行點後再繼續。



下載 cuDNN

cuDNN 是 NVIDIA 開發用來為深度神經網絡進行 GPU 加速的函式庫，cuDNN 可支援通用的 AI 學習框架，如：Tensorflow、Pytorch、caffe 等等。

cuDNN 的主要特性

- ✓ 為各種常用卷積實現了 Tensor Core 加速，包括 2D 卷積、3D 卷積、分組卷積、深度可分離卷積以及包含 NHWC 和 NCHW 輸入及輸出的擴張卷積
- ✓ 為許多電腦視覺和語音模型優化了內核，包括 ResNet、ResNext、

EfficientNet、EfficientDet、SSD、MaskRCNN、Unet、VNet、BERT、GPT-2、Tacotron2 和 WaveGlow

- ✓ 支援 FP32、FP16、BF16 和 TF32 浮點格式以及 INT8 和 UINT8 整數格式
- ✓ 4D 張量的任意維排序、跨步和子區域意味著可輕鬆整合到任意神經網路實作中
- ✓ 能為各種 CNN 體系架構上的融合運算提速
- ✓

NVIDIA cuDNN is a GPU-accelerated library of primitives for deep neural networks.

Download cuDNN v8.9.7 (December 5th, 2023), for CUDA 12.x

Download cuDNN v8.9.7 (December 5th, 2023), for CUDA 11.x

Local Installers for Windows and Linux, Ubuntu(x86_64, armsbsa)

Local Installer for Windows (Zip)

Local Installer for Linux x86_64 (Tar)

Local Installer for Linux PPC (Tar)

Local Installer for Linux SBSA (Tar)

Local Installer for Debian 11 (Deb)

Local Installer for Ubuntu18.04 x86_64 (Deb)

Local Installer for Ubuntu20.04 x86_64 (Deb)

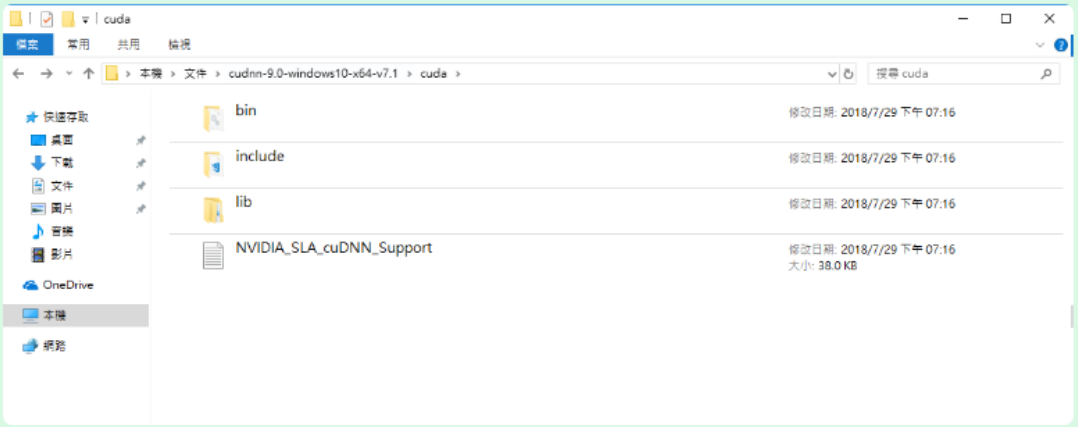
Local Installer for Ubuntu22.04 x86_64 (Deb)

Local Installer for Ubuntu20.04 aarch64sbsa (Deb)

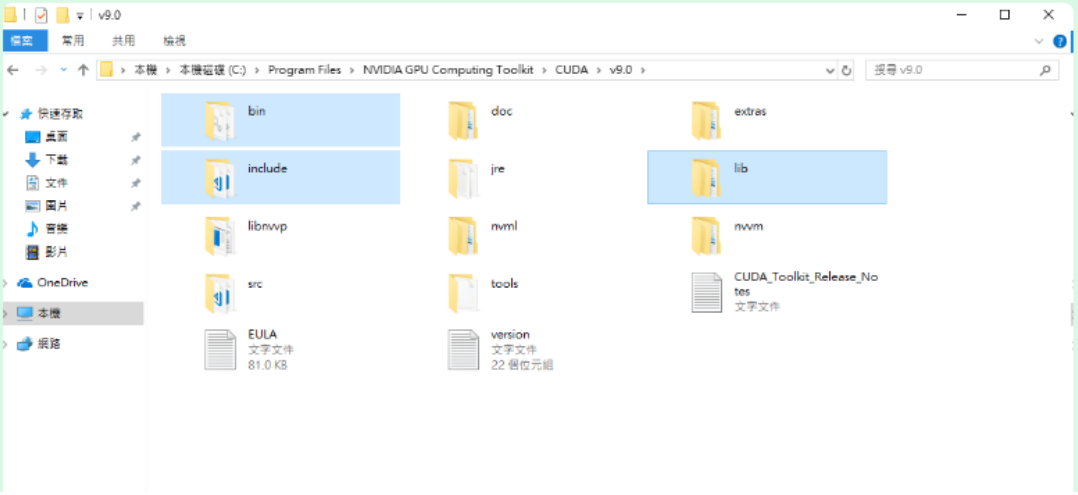
Local Installer for Ubuntu22.04 aarch64sbsa (Deb)

Local Installer for Ubuntu20.04 cross-sbsa (Deb)

之後會得到一個 CUDA 資料夾，分別含有 `bin` 、 `include` 、 `lib` 三個資料夾



將其內部的檔案，分別移至 `C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v9.0` 路徑下的對應資料夾中



打開控制台→系統及安全性→進階系統設定→進階→環境變數（或是直接在控制台中搜尋 PATH）

尋找「系統變數」中「Path」的部份並用左鍵雙擊，新增下述變數：

- C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v8.0\bin
- C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v8.0\lib\x64