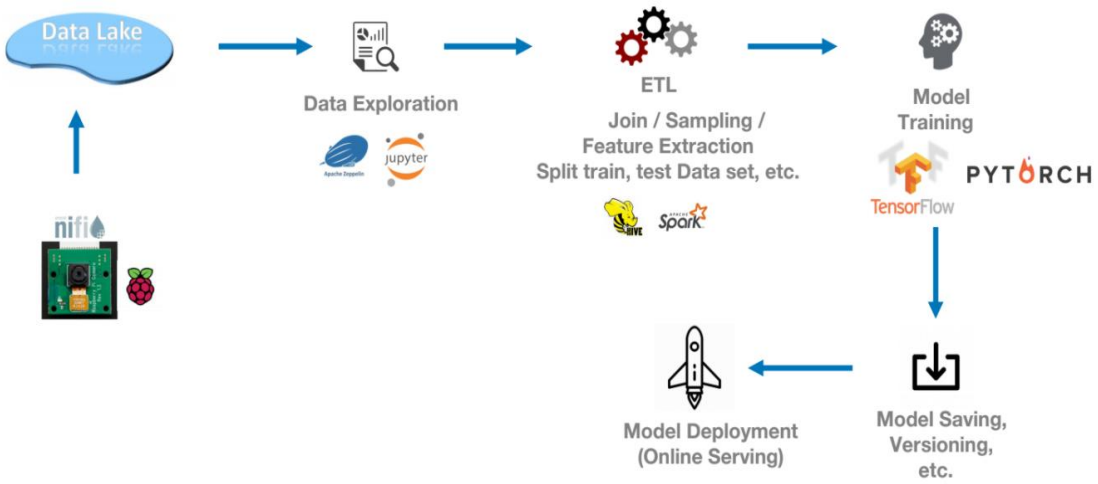
2020_OS_Fall_HW4: MLOps

學號：F14076083
姓名：魏湧致
系級：資訊 111

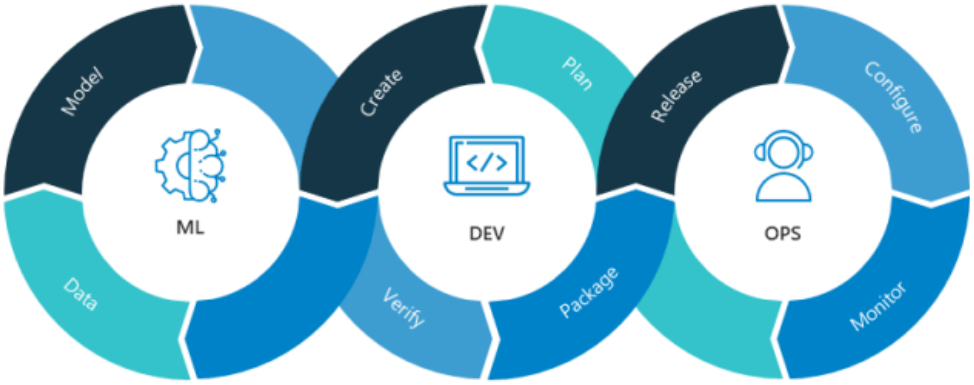
- Machine Learning Workflow



1. 收集資料 (Gathering Data)
   收集訓練需要用到的所有資料放入 Data Lake 內。
2. 資料探索 (Data Exploration)
   透過工具對資料進行初步的處理，了解資料的基本結構。
3. ETL (Extract-Transform-Load)
   從原本的 raw data 取出乾淨的 data 以及 data 的 feature，才能夠進行後續的 Training。
4. 訓練模型 (Model Training)
   選擇合適的模型後利用 TensorFlow, PyTorch 等工具來進行訓練，透過不斷的訓練來調整模型的參數以達到我們希望的結果。
5. 儲存模型與版本控制 (Model Saving, Versioning)
   在訓練的過程中可能因為 Data Set 的改變使得結果不同，這時候就要利用版本控制的方式來確保模型的正確性。
6. 模型發布 (Model Deployment/Serving)
   完成訓練後就能夠將這個模型實際的應用。

- MLOPs
  現在 Machine Learning Dataset 的 size 越來越大，專案的參與人員增加，結合 MLOps 的方式進行開發變成必要的選擇，如下圖一[1]，MLOps 結合了

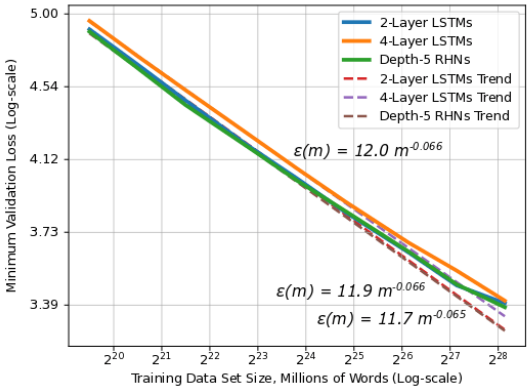
machine learning 與 DevOps，除了 DevOps 所涵蓋的版本控管 (Source Control)、持續整合 (Continuous integration)、持續交付(Continuous Delivery)、持續部署 (Continuous Deploy)，MLOps 更多了資料集處理 (DataSet)、Machine Learning (Model training, Algorithms) 等元素，對 MLOps 流程的優化將在以下幾點與結語提到。



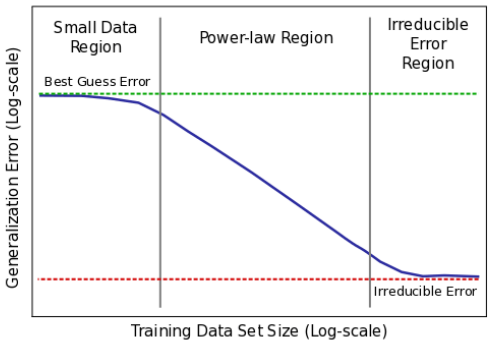
圖一

- 檔案系統(File System)

  在[2]的影片裡 TensorFlow 的工程師提到分散式檔案系統對現今的 Machine Learning 是很重要的，如下圖二、三[2]，Data Set 的大小與 Model 準確率的提高有著密切的關係，因此能夠將大量的資料分散到各個 node 中的分散式檔案系統，對於存放這類大規模的數據集是較合適的選擇，也能夠更有效的讓系統的所有使用者共享資料，不用每個使用者都複製一份 Data Set。



圖二



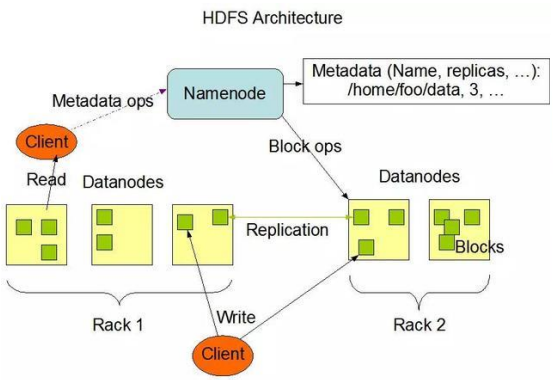
圖三

使用 Hadoop distributed file system (HDFS)加上額外的功能

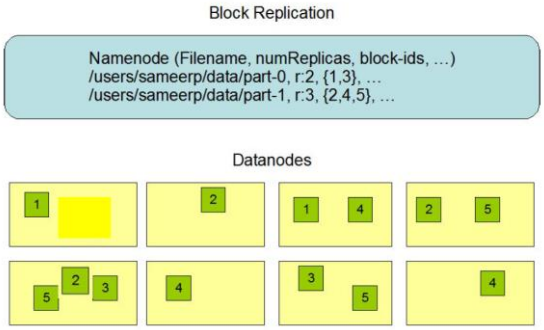
- 如下圖四，一個 File 會被分成多個區塊存放在多個 Datanodes 上，這樣在 Training Model 時就不需要讀取全部的檔案，適合 SGD 這種一次

操作一個 sample 的方式，而如果要讀取多個 block 也能夠平行讀取加快速度。

- Datanodes 傳送 heartbeat 給 Namenode 的機制能夠偵測 datanodes 是否有問題以恢復資料，確保 data set 沒有遺失或缺失。
- 因為對於資料進行修改會記錄在 Namenode 的 EditLog 中，當資料的分布改變或有所增減時就能夠另外再提醒使用者必須重新訓練模型，或在有閒置資源時自動重新建模，再通知使用者確認結果是否有更好，另外 System 也能自動檢測 Data 中的極端值提醒使用者，以增加後續 training 的準確率。
- 訓練完成後模型的存放因為 HDFS 會複製多份資料備份(圖五)，所以能夠確保模型的保存。另外在訓練的過程中會不斷地調整參數與資料集，除了能加上 git 來進行版本控制，紀錄每次 release 的 model，System 也能自己記錄每次的訓練是用到哪些 data、哪些參數以及資源的使用率，以方便使用者若覺得前幾次的版本較好能夠查看。



圖四



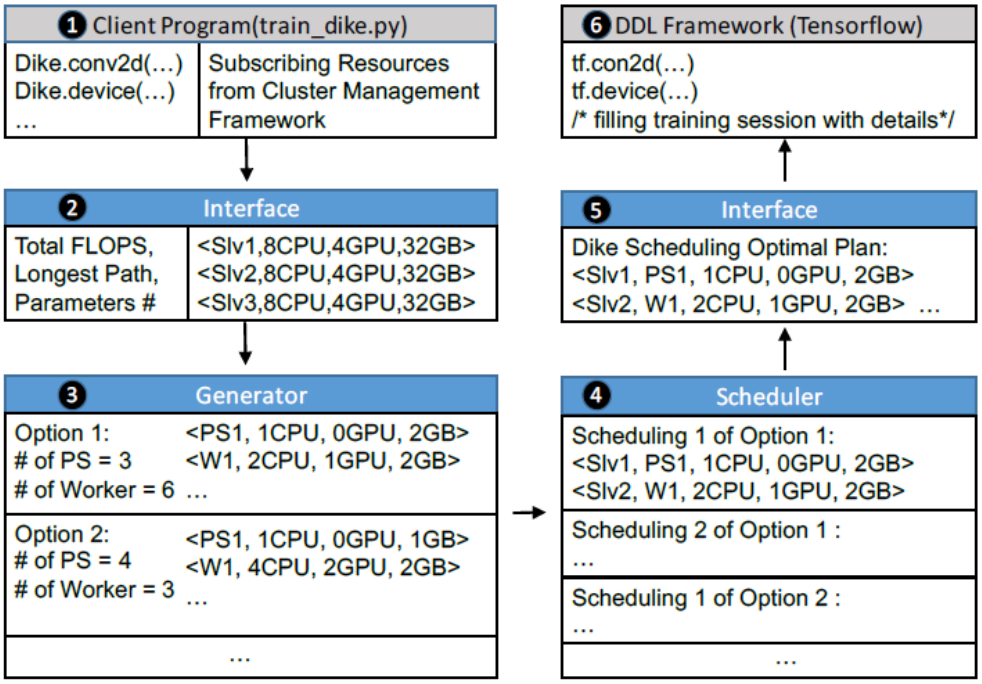
圖五

- 資源配置(Resource Management) 與 Process 排程(Scheduling)

在 Machine Learning 中選擇不同的模型與不同的訓練方式所需要使用到的資源都有所不同，因此我們可以如下圖六[4]在程式 run time 時使用 Generator，根據使用者的模型與參數來動態的分配 CPU、GPU、memory 等資源，這些資源分配的選擇會再傳送給 Scheduler 來排程。

- 因為需要將每一 batch 的 Parameter 放到 memory 內，參數的數量可以用來預估 memory 的使用量，另外因為是分散式的系統，也能預估網路的流量。
- 在進行 gradient descent 時有使用到的參數可以幫我們預測會有多少的 float point computation(FLOPs)，而算出 FLOPs 就能計算出總計算時間，來幫助我們在配置 CPU 與 GPU 時取得較好的平衡，例如能夠讓 CPU 來準備資料，GPU 來進行運算，另外也能夠避免被分配到的資源沒有適當的利用。

- 雖然 Scheduling 時將各個 resource assignment 分配到 nodes 上是 Bin-Packing 的 NP 問題，但因為 Generator 只產生有限的 task option，我們可以使用 Next Fit Algorithm 來解決。
- 在 locality 的部分，較小的 datasets(如 Cifar10)因為能夠將資料完整的放到 memory，因此較無影響，較大的 datasets(如 Imagent)，memory 是用來 cache input，故只會對 PS(更新參數)這部分有些許影響。



Detailed process of Dike.

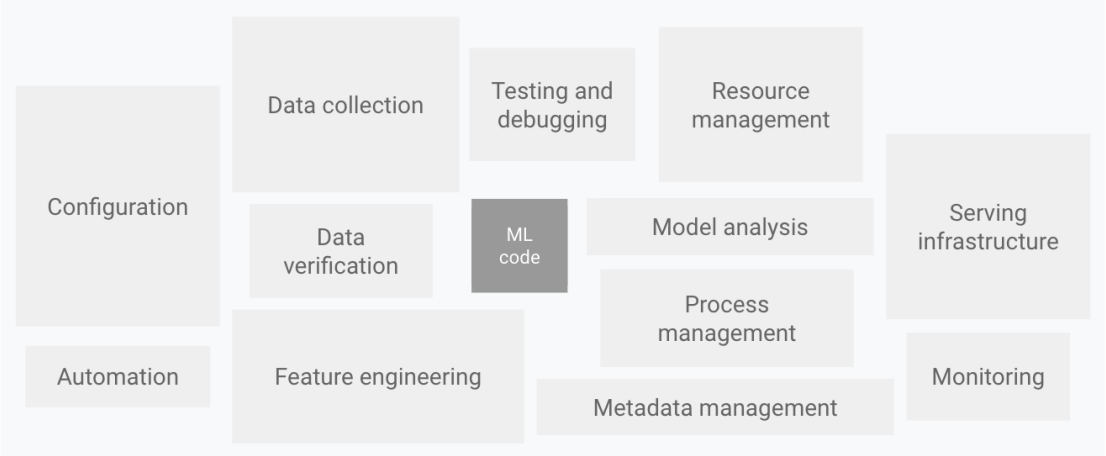
圖六

- 結語

設計的 OS 對 MLOps 有所幫助的部分如下

- DataSet Sources (大資料的儲存、資料分布的檢測)
- Training Model 版本控管與自動訓練
- Training Model History
- Training Model 的可回溯性
- 分散系統中資源的配置與排程

如圖七[5]，Machine Learning Code 只佔了整個機器學習架構的一小部分，而參與的人員包括資料科學家、資料工程師等對系統的底層並不是很了解，這時候就需要有一個好的平台來讓整個 ML 的團隊彼此溝通協調、讓 MLOps 的流程更順暢運作，而透上述的這些方法我認為能夠幫助 Machine Learning 流程。



圖七

- 參考資料

[1] https://reurl.cc/9XdO8n

[2]https://www.youtube.com/watch?t=13m59s&v=SxOsJPaxHME&feature=youtu.be&ab_channel=TensorFlow

[3] http://research.baidu.com/Blog/index-view?id=89

[4] https://www.mdpi.com/2079-9292/8/3/327/pdf

[5] https://cloud.google.com/solutions/machine-learning/mlops-continuous-delivery-and-automation-pipelines-in-machine-learning