科技部補助專題研究計畫成果報告

（期中進度報告）

以區塊鍊及深度神經網路設計霧運算物聯網之分散式安全

Securing Fog-based IoT Systems using Blockchain and Deep Neural Networks

計畫類別：🗹個別型計畫 □整合型計畫

計畫編號：MOST 107-2221-E-194-001-MY3

執行期間： 107年 8月 1日至 110年 7月 31日

執行機構及系所： 國立中正大學資訊工程學系

計畫主持人： 熊博安 教授

共同主持人：

計畫參與人員： 蕭諭謙、簡毅、林育聖、李冠宏、林柏儒、邱皇旗、張譯予、林岑叡、Darmawan Utomo, Than Dao, Sapdo Utomo, Thanh Tung

本計畫除繳交成果報告外，另含下列出國報告，共 1 份：

□執行國際合作與移地研究心得報告

☑️出席國際學術會議心得報告

□出國參訪及考察心得報告

中 華 民 國 108年 5月 31日

中文摘要

本計畫目標為以區塊鏈及深度神經網路設計技術強化基於霧運算之物聯網分散式安全。具體而言，計畫最終目標是完成一個智能物聯網閘道器(AIoT Smart Gateway)。第一年我們已經完成AIoT閘道器之架構設計，包含動態模組化之容器(container)設計、容器中長短記憶(Long-Short Term Memory, LSTM)異常偵測模型設計、容器中區塊鏈設計等。計畫的實驗環境為Raspberry Pi。以下說明此架構的三大特色功能包含使用容器技術達到閘道器模組化架構設計、使用LSTM深度神經網路模型進行異常偵測及使用區塊鏈技術進行終端分散式安全設計。

第一個部分有關 "閘道器設計模組化"，本計畫採用輕量級虛擬化技術 Docker，並且將Docker實作到Raspberry Pi 平台上，使異常偵測的LSTM模型和區塊鏈均能在各自的Docker Container中獨立運行。落實Docker容器技術在終端裝置也是本計畫的特色之一，因為這樣我們的物聯網閘道器就是一個具有自動遠端部屬功能的設計。第二個部分有關 "異常偵測" (anomaly detection)，目前計畫第一年已經在Raspberry Pi的平台上實作完成200-Blocks的LSTM模型設計。此模型可以在終端非常有限的資源中，進行大量智慧電表資料的異常偵測。模型先在雲端進行訓練(training)後，在終端平台進行推論 (inference)。目前的終端模型版本，啟動時大約需要57秒(下載深度學習Keras, Tensorflow框架、模型權重等)，而偵測200個電表的資料有無異常一次僅需947us，亦即每半小時的電表資料只需花947us即可偵測異常。第三個部分有關 "物聯網資料分散式安全設計"，計畫中已經完成在Raspberry Pi的區塊鏈設計。目前的設計包含兩個組織(organizations)，每個組織包含兩個節點(peers)，一個負責驗證，一個負責交易，負責發起交易的節點透過智能合約(smart contract)發起交易，並且維護帳本(ledger)的orderer節點負責確保兩邊的帳本是一致的。終端區塊鏈的設計是本計畫的創舉。

綜合上述三大特色功能，本計畫在第一年已經完成第一個版本的AIoT Smart Gateway，包含自動部屬的容器設計、LSTM的異常偵測模型以及使用區塊鏈的資料保護。這樣的AIoT閘道器設計，目前是世界第一，尚未在其他計畫或學校看到。未來第二年，我們將此架構設計優化，包含各容器之間的協調、資料統計、雲端服務的整合等。第三年，我們將此閘道器設計應用到其他領域。

關鍵詞：霧運算、輕量級虛擬化技術、異常偵測、長短期記憶模型、區塊鏈

英文摘要

The main goal of this project is to enhance the security of Fog-Based IoT Systems using Blockchain and Deep Neural Networks. In more concrete terms, the project’s final goal is to design and implement an AIoT Smart Gateway. In the first year of the project, we have already completed the AIoT’s architecture design, including dynamic modularization via container design, LSTM anomaly detection model in container, and blockchain design in container. We have implemented the design in a Raspberry Pi platform. This architecture now consists of the following three innovative designs including container-based modular design, LSTM model-based anomaly detection at the edge, and blockchain-based data protection at the edge.

First about "modular gateway design", we have implemented the light-weight virtualization technique Docker containers on Raspberry Pi platform, so that the LSTM-based anomaly detection model and the blockchain-based data security module can execute independently in their own respective containers. Realizing docker containers at the edge device is also one of this project’s innovations because the functionality in our AIoT smart gateway can be automatically and dynamically deployed.

Second about "IoT edge-based anomaly detection", in the first year the project team has already implemented a 200-block LSTM model design on Raspberry Pi platform. This model can perform anomaly detection of a large amount of data in highly resource-constrained edge devices. The model is first trained in the cloud and then installed into a container at the edge for inference. This edge detection model need 57 seconds for initialization (downloading of Keras and Tensorflow frameworks and model weights) and then it only requires 947 us for anomaly detection for one set of 200 smart meter data, collected in a 30-minute timestep.

Third about "IoT data distributed security", we have already implemented a simple blockchain system on Raspberry Pi. This system consists of 2 organizations, each of which has 2 peers, one for verification and one for transaction processing. The latter peer starts a transaction via smart contract design, which is then verified by the former peer. The orderer is responsible for maintaining consistency between the 2 versions of the ledger in the 2 organizations. Edge-based blockchain system design is one of our main innovations in this project.

Summarizing the above three innovative functions, in the first year of this project we have already implemented the first version of the AIoT Smart Gateway design, which includes automatic dynamic deployment via container design, LSTM-based anomaly detection model, and blockchain-based data security design. This AIoT smart gateway design is the first of its kind in the world, as yet unseen in the literature or other sources. In the second year, we plan to optimize the gateway design by creating the automatic coordination among containers, data statistics collection, integration with cloud services, etc. In the third year, we plan to apply this AIoT Smart Gateway design to other application domains.

Keywords : fog computing, lightweight virtualization technology, anomaly detection, long-short term memory (LSTM), blockchain

目錄

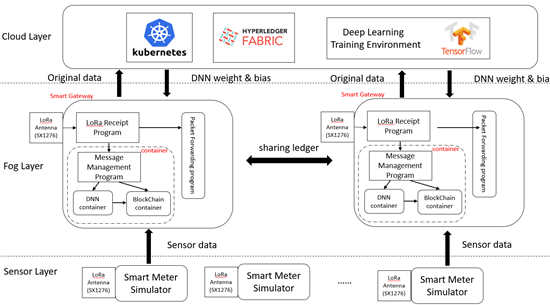
1. 研究目標
   1. 基於霧運算之物聯網架構
   2. 智慧閘道器設計目標
   3. 輕量級虛擬化技術
   4. 深度異常偵測模型
   5. 區塊鏈之安全架構
2. 研究方法
   1. 智慧閘道器架構設計
   2. LoRa通訊模組設計
   3. Docker容器設計與管理
   4. LSTM異常偵測模組設計
   5. 區塊鏈模組設計
3. 實驗結果
4. 文獻探討
5. 研究目標

本計畫的目標主要為基於物運算之物聯網智慧閘道器(Fog-based IoT Smart Gateway)設計，包含三大特色(1)動態模組化與自動部屬、(2)終端異常偵測及(3)終端資料分散式安全。以下先說明Fog-based IoT架構，再說明在三大目標下本計畫希望達成之各項功能。

在智慧閘道器中，首先以Docker容器技術落實到終端，以達成動態模組化與自動部屬。第二，以終端深度神經網路模型，達成在終端進行異常偵測。最後，以區塊鏈落實到終端，以達成物聯網資料分散式安全設計。

# 基於霧運算之物聯網架構

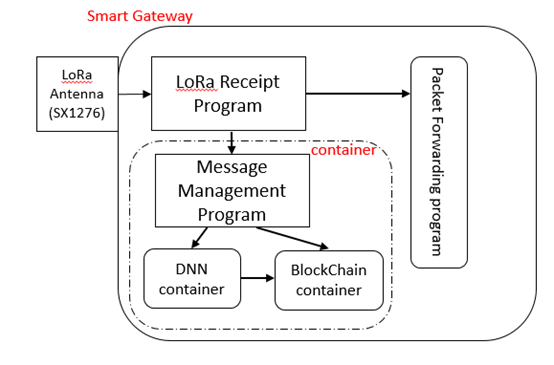
本計畫的目標智慧電網之物聯網(smart grid IoT)架構，如圖一所示可分為三個層級，亦即感應層(Sensor Layer)、霧運算層(Fog Layer)及雲計算層(Cloud Layer)。在感應層中，經由LoRa無線網路模組取得智慧電表的用電資料，再藉由霧運算層智慧閘道器中的深度神經網路模型分析並偵測目前智慧電表之用電記錄是否正常(如預期)[5]。若有偵測到異常，將進一步分析異常狀況例如統計異常次數等。偵測結果都會被記錄在區塊鏈的帳本中，作安全的管理，並且將這些資訊上傳到雲計算層，以便作為後續模型重新訓練(re-training)的數據資料。



圖一、以區塊鏈及深度神經網路設計霧運算物聯網架構

# 智慧閘道器設計目標

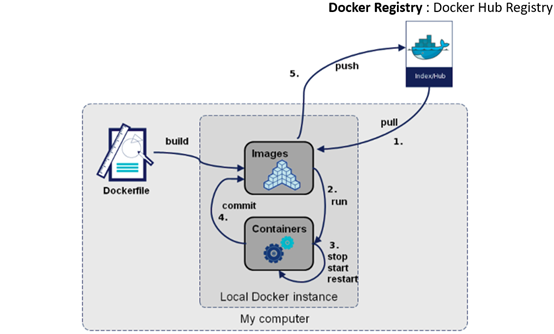
本計畫第一年所設計的霧運算運層智慧閘道器，如圖二所示，具有LoRa封包接收、訊息管理、兩種容器及封包傳遞。容器分為兩種，亦即DNN container及BlockChain container。前者包含LSTM異常偵測模型設計。後者包含區塊鏈系統，協助儲存與保護用電記錄。



圖二、智慧閘道器(Smart Gateway)架構

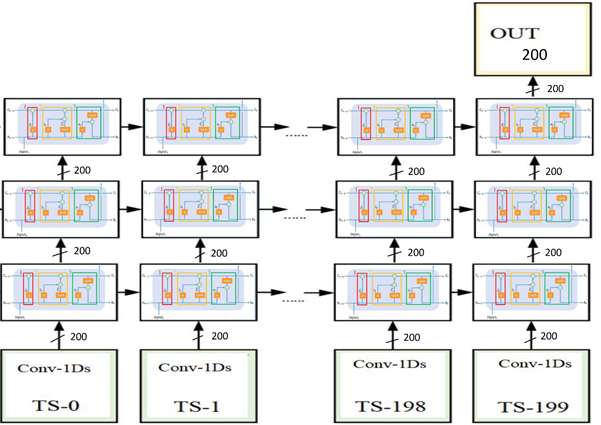
# 輕量級虛擬化技術

Docker是一個開源專案，如圖三所示包含三個主要部分:Docker映像檔(Image)、Docker容器(Containers)及Docker註冊伺服器(Registries)。映像檔是一個唯讀的環境模板，裡面包含了容器內所有的程式(包括應用程式、相關函式庫、設定檔)，可以用來建立容器，我們撰寫Dockerfile來建立符合我們系統所需要環境的映像檔；Docker容器是根據映像檔建立的。映像檔是唯讀的，而容器則是映像檔的可寫層，每個容器都是相互獨立的；Docker註冊伺服器上有許多倉庫(repository)，可分為公開和私有，倉庫內存有很多映像檔可以供人使用。本計畫第一年已經完成在Raspberry Pi終端平台上的Docker設計。

圖三、Docker Flow [18]

# 深度異常偵測模型

終端異常偵測之目標，擬在Docker容器中實作一個200單元的LSTM模型，其可以根據過去200筆歷史資料預測下一時段(timestep)的用電資料[2]。如圖四所示，目標模型包含三層的LSTM blocks[3]。

****

圖四、具有200單元的LSTM異常偵測模型

# 區塊鏈之安全架構

## 在本計畫第一年目標之一，在終端容器中落實區塊鏈系統的設計[4]。以下先說明區塊鏈的特性與架構。

## 區塊鏈的特性

本計畫的目標之一為在Docker 容器中實作區塊鏈分散式帳本(Decentralized Ledger)。每個Docker 容器記錄區塊鏈分散式帳本的狀態，而每個帳本會記錄過去所有交易以及更改方式，將資料加密及分散式儲存，使資料有更安全的保護，包含資料完整性與交易完整性。

* 資料完整性

當某一個節點提出交易時，需要其他的節點去認證他的交易[1]。由於區塊鏈的安全是建立在許多的節點之上，越多節點代表越多的分散式帳本，資料越安全。

* 交易完整性

區塊鏈的特色是擁有智能合約，可以用來描述多個組織之間所做的決策。智能合約具有不可更變性，可有效保護交易的完整性，因此適用於商業交易。

## 區塊鏈架構中所參與五個主要成員

* + - 節點(Node)：我們將節點定義為對等節點。在一個設備中有4個節點。節點是區塊鏈中的通信實體。不同類型的多個節點可以在同一物理服務器上運行。重要的是節點如何在區塊鏈的系統中分組並且控制它們。有三種類型的節點：(1) 客戶端節點：向區塊鏈系統提交實際交易調用的客戶端，並將交易傳送到排序節點服務。(2)對等節點：提交交易並維護狀態和分類帳副本的節點。(3)排序節點：運行通信服務的節點，實現交付保證，並且排序交易。
    - 對等體(Peer)：用來存放分類帳和智能合約，是Hyperledger Fabric網路中的基礎結構。
    - 帳本(Ledger)：區塊鏈的核心便是分散式的分類帳，它用來記錄區塊鏈網路上所發生的所有事物。區塊鏈的分類帳通常被當作是分散的，因為它會被複製到許多的網路參與者，也就是Peers之中。讓每一位參與者都在裡面進行維護和協作。 Hyperledger Fabric有一個分類帳子系統，包括兩個組件：World State 和Transaction Log。每個參與者都擁有他們所屬的分類帳本。World State描述了在給定時間點的分類帳的狀態。它是分類帳的數據庫。Transaction Log記錄World State當前值的所有交易。分類帳就是World State和Transaction Log歷史記錄的組合。分類帳中Ｗorld State是可以替換的。預設的情況下，Hyperledger Fabric使用LevelDB當作預設的資料庫。Transaction Log是不可以更改的。它只記錄區塊鍊網絡使用的分類帳資料庫的前後值。
    - 智能合約(Chaincode)：用來定義資產或資料的軟體，裡面包含了修改資產或其他交易的指令。此為區塊鏈中的邏輯。智能合約可以執行讀取或插入等功能，用以改變資料庫的訊息或規則。
    - 排序模塊(Orderer)：將交易的資料順序委託給Orderer去排序以達成共識。確保區塊鏈的交易不會進行分叉或是讓他人有機會加入其他資料進去。

## Hyperledger Fabric

* + - Cryptogen tool：幫助網路實體生成加密的證書和簽名密鑰，如圖五，這些證書就像身份一樣可以允許在交易時進行簽名和驗證，Cryptogen工具所使用的文件為 crypto-config.yaml，此檔案內容包含網路拓墣，且允許我們幫每個組織和節點生成一組證書(certificate)和密鑰(key)。每個組織都配置為一個根證書(ca-cert)，可以將特定的組件綁定到該組織，接著將為每個組織產生專屬CA證書(具有排他性)。



圖五、使用Cryptogen為區塊鏈的成員生成證書和密鑰

* + - configtxgen tool:用於創建三個配置的工作排序節點的創世區塊通道配置交易錨節點，對應到組織，用來互相溝通使用，此工具使用的文件為configtx.yaml。

1. 研究方法

# 智慧閘道器架構設計

本計畫第一年已經完成在Raspberry Pi平台上一個完整的智慧閘道器設計，其中包含LoRa無線通訊模組、兩種Docker容器與管理包含LSTM異常偵測模型及區塊鏈系統、封包傳送等。智慧閘道器的整體架構設計如圖二所示。本節後面將詳細說明上述各個部份。

## LoRa 無線通訊模組設計

## 智慧閘道器中，LoRa無線通訊模組主要負責LoRa封包的接收與發送[6,7,9]。如圖六所示，我們採用SX1276收發模組及天線接收LoRa的封包。目前的設計支援收發200個智慧電網的用電資料。每半小時(half hour timestep)，每一個智慧電表將送ID, timestamp, electricity usage等資料給閘道器。閘道器收到智慧電表資料後，即發送個別資料給總管Docker Container。此總管容器負責管理容器之間的通訊，因此可讓智慧電表資料傳送至LSTM異常偵測模組的容器，進行異常偵測。偵測結果 (0 = 無異常、1 = 異常) 將與資料一起送至區塊鏈的容器中。

## 實驗中，如圖六所示，感測節點與閘道器，各採用一Raspberry Pi 3開發板及SX1276 LoRa無線傳輸模組作為開發環境。

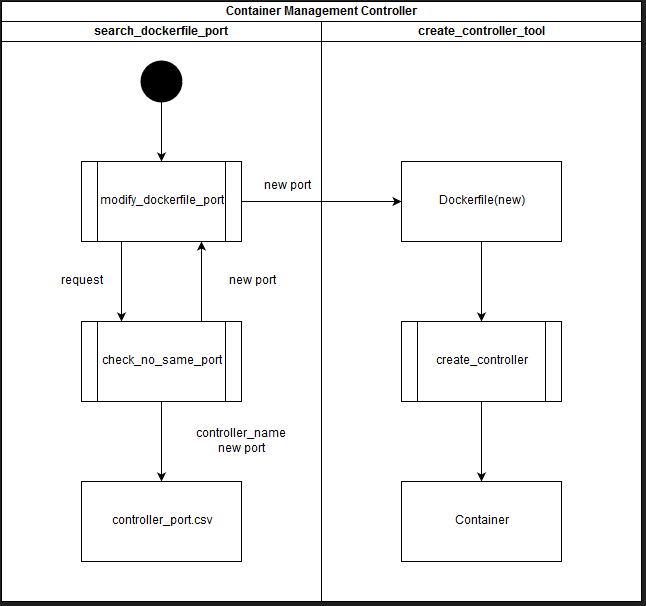
## 

## 圖六、SX1276 LoRa Module

## Docker容器設計與管理

* + 1. Docker 容器設計

在本計畫的智慧閘道器設計中，已導入輕量級虛擬化技術(Docker Container)。容器技術使智慧閘道器可快速且自動佈署相關功能，例如異常偵測及區塊鏈分散式帳本。除了功能容器外，智慧閘道器中亦建立一個管理的容器，負責容器之間的通訊。在管理容器裡，包含docker\_container\_tool.py 的程序，當管理容器接受到新的資料時，會自動呼叫 Docker 腳本。此腳本將查詢現有的阜 ( port )，將尚未使用的阜，分配並且創建新的容器使用。圖七為腳本執行的流程。



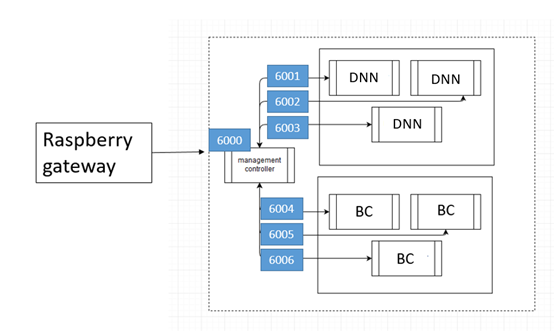
圖七、容器管理控制器流程

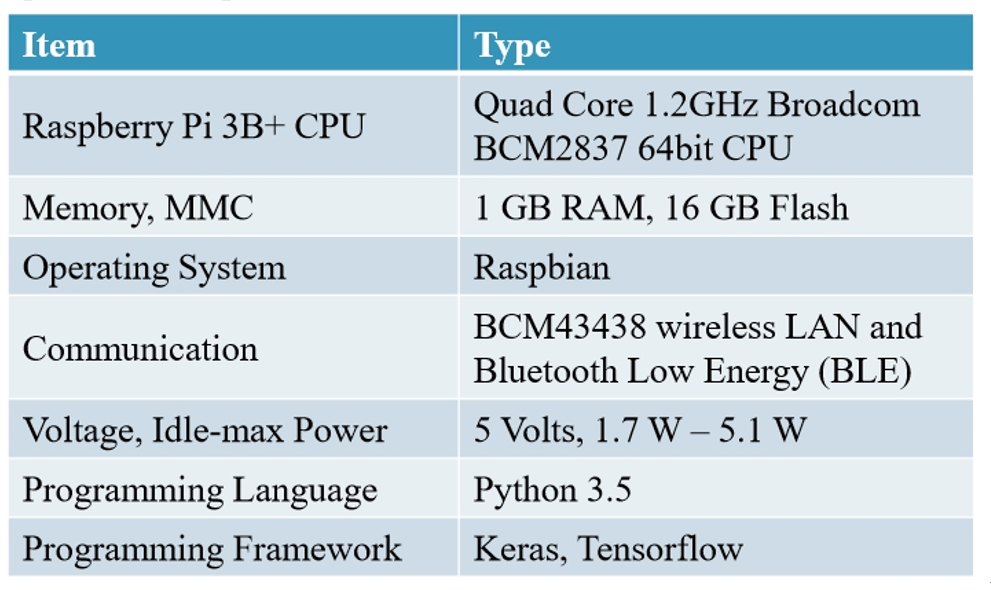
* + 1. 容器管理

在智慧閘道器中，為避免每個容器間互相建立各自的通道，而造成阜號重覆以及IP過多的問題，本計畫提出額外建立一個內部通訊網路。此內部網路，將每個容器的IP或是網域(Domain)，與管理容器共用相同的網路服務，使每個容器間只要透過相同的IP，以阜號為傳輸目標，即可達到資訊的交流。內部網路架構如圖八所示。6000阜為管理容器之通訊阜，而6001~6003為LSTM異常偵測模型容器之通訊阜，6004~6006為區塊鏈容器之通訊阜。

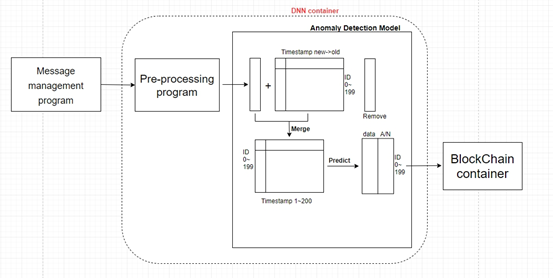
# LSTM異常偵測模型設計

為了配合智慧閘道器中容器的設計，LSTM異常偵測模型亦需要設計到容器之中。計畫的實驗環境如圖九所示。第一年，如圖十所示，我們已經成功實作出在容器中的LSTM異常偵測模型。此模型在雲計算曾進行訓練，並且在感測層的終端裝置(Raspberry Pi)進行推論(inference)亦即異常偵測。計畫第一年我們參考的智慧電表資料(<https://combed.github.io/>)，使用其中160,000筆資料。以3:1的比例，將智慧電表資料分為訓練及測試的資料集(datasets)。容器中的LSTM模型，需要一個前置處理程式(pre-processing program)主要負責將從管理容器收到的智慧電表即時資料進行排序。排序完的多個電表一個timestep資料將進行time window 整理，也就是刪除最舊的資料，加入最新的資料。再將此整理過的 time window 資料，餵給LSTM模型進行下個時段的預測。預測的結果，將附加於資料後面。整併在一起的電表資料與異常偵測結果，將送至區塊鍊的容器中。

圖八、Docker Management



圖九、實驗環境



圖十、在容器內LSTM異常偵測模型架購

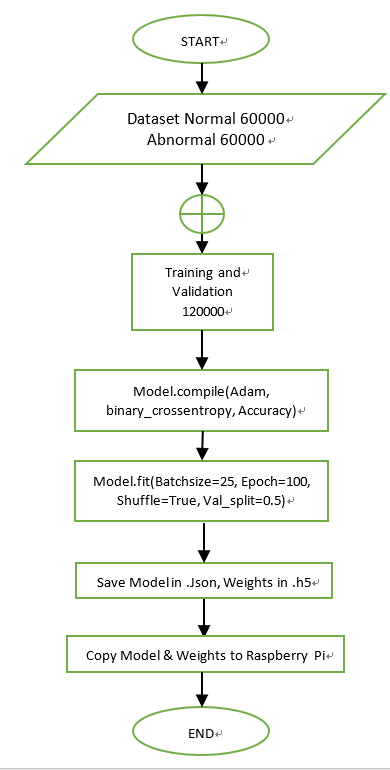
1. LSTM模型訓練 (Model Training)

圖十一為LSTM模型的訓練流程，相關逐步如面所說明。

Step1: 將正常資料集(Normal Dataset)與異常集(Abnormal Dataset)，進行標準化(Normalize) ，使數值介於 0 到 1 之間。

Step2: 將資料打散，放入model做訓練，模型上以分批(batch)的方式，放入訓練model。

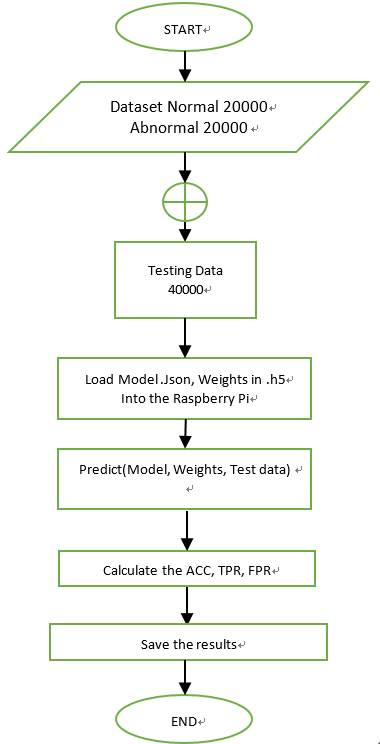
Step3: 將訓練好的模型，複製到Raspberry上。



圖十一、 LSTM模型訓練流程

1. LSTM模型測試 (Model Testing)

圖十二為LSTM模型的測試流程，相關逐步如後面所說明。

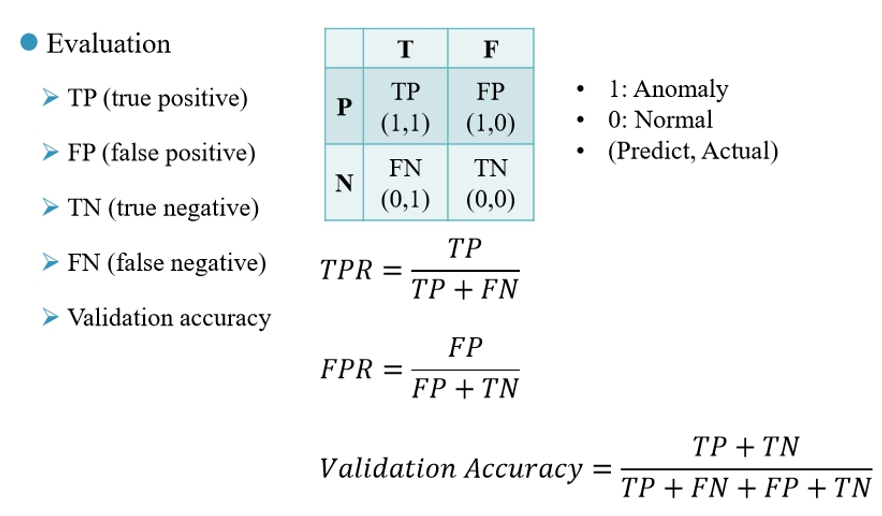


圖十二、 LSTM模型測試流程

Step1: 將正常資料集(Normal Dataset)與異常集(Abnormal Dataset)，進行標準化(Normalize) ，使數值介於 0 到 1 之間。

Step2: 將資料打散，放入model進行訓練。

Step3: 將預測的結果，使用混淆矩陣(Confusion Matrix)作為評估(Evaluate)標準。相關作法如圖十三所示。



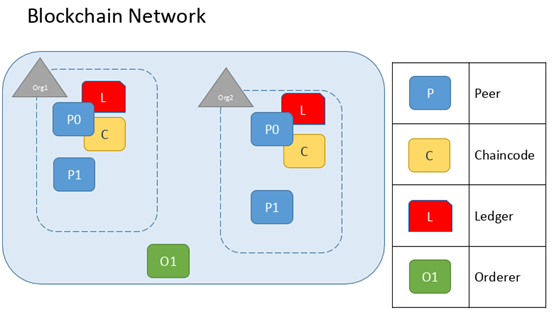
圖十三、評估標準-混淆矩陣

# 區塊鏈模組設計

## 本計畫第一年主要完成終端裝置容器中區塊鏈的設計。其應用很廣，例如可用在Carpooling [10]。

## 架構說明

如圖十四所示，我們所設計的終端區塊鏈包含兩個組織Org1, Org2。每個組織包含兩個節點，一個負責驗證(P1)，一個負責交易(P0)，負責發起交易的節點P0透過智能合約(Chaincode)發起交易(transaction)，並且維護帳本(L)的orderer節點(O1)，負責確保兩邊的帳本(L)是一致的。



圖十四、 區塊鏈的架構

## 交易查詢

當客戶利用應用程序查詢資料時，Peer可以立即的把查詢結果回應給客戶端，因為所有資料都會在Peer的分類帳的副本之中，Peer也不需要等待其他Peer回應客戶端的查詢，換句話說不會浪費時間等待，達到減少延遲的時間，客戶端可以連接到不同的Peer進行查詢，以便查詢不同資料。

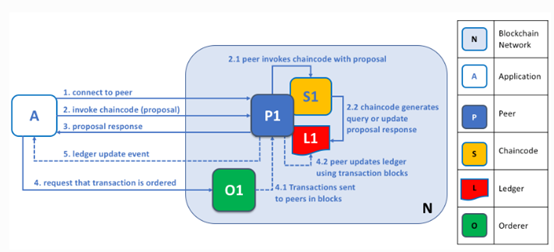
由圖十五可瞭解應用程式如何管理區塊鏈內部的數據。Orderer節點O1負責確保分類帳在各個其他節點中保持同步。在區塊鏈中完成一個交易，有可以列出以下主要步驟：

1. 應用程序A連接到Peer P1。

2. Peer P1可以調用智能合約S1來查詢或是更新分類帳L1。

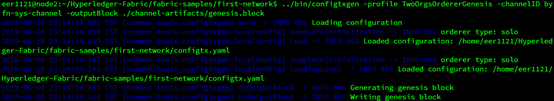
3. 應用程序A接收回應。

4. 應用程序A將交易傳送至排序節點O1進行排序。並且蒐集其他交易到區塊之中，把交易分發給所有的Peers，更新帳本L1。

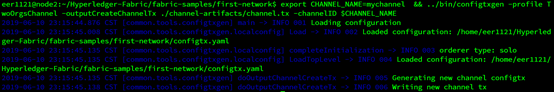
圖十五、區塊鏈內部的架構和交互流程

## 實作方式

* 在實作過程中，透過 configtx.yaml，模擬三個成員，一個負責排序服務的組織OrdererOrg和兩個節點組織Org1及Org2。每個組織皆管理和持有2個對等節點peer0和peer1。此文件中也配置錨節點(peer0.org1.example.com和peer0.org2.example.com)，負責用於兩個組織之間的溝通，其次也為每個成員指定MSP文件夾，用來指定數字簽名和進行驗證，並且創建頻道配置，如圖十六所示。$CHANNEL\_NAME或者將CHANNEL\_NAME設置為整個說明中可以使用的環境變量，如圖十七所示。

****

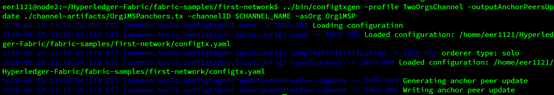
圖十六、使用Configtxgen 工具創建創事區塊和配置通道

****

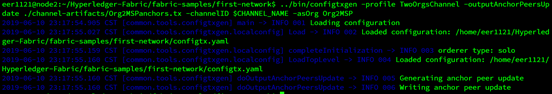
圖十七、將頻道名稱設置為環境變量

接下來，將正在構建的通道上定義組織一的錨節點並且再次確認

$ CHANNEL\_NAME已被替換或者為以下命令設置了環境變量，如圖十八與圖十九:

****

圖十八、定義組織一的錨節點

****

圖十九、定義組織二的錨節點

配置完成後，如圖二十所示，我們使用Docker-compose腳本啟動區塊鏈網路，協助快速啟動Docker容器，並且進入docker容器，變更環境變數。如圖二十一所示，將剛剛創立的mychannel通道加入到orderer節點之上。如圖二十二所示，再將mychannel通道加入組織的peer上，讓orderer節點和組織一能夠藉由此通道相互溝通。

****

圖二十、變更環境變量進入組織一的peer0

****

圖二十一、將mychannel通道入到orderer節點

****

圖二十二、將mychannel通道加入peer0.org1.example.com

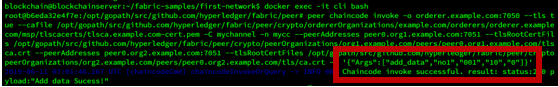
加入第二個組織，如圖二十三所示，首先必須先將mychannel通道加入組織二的peer0上，讓orderer節點，組織一和組織二能夠藉由此通道相互溝通。

****

圖二十三、將mychannel通道加入peer0.org2.example.com

上述的區塊鏈網路架構創建完成也啟動後，我們開始撰寫及安裝智能合約，首先我們調用智能合約中的 Invoke function 將資料儲存至帳本當中，如圖二十四的紅框處介紹我們儲存的資料屬性如下[8]:

1. 智慧電表名稱(Smart meter Name)
2. 智慧電表編號(Smart meter ID)
3. 智慧電表記錄的用電量(Electricity Charge)
4. 用電資料經由LSTM模型判斷是否正常，0為正常，1為不正常(Normal/Abnormal Information)

****

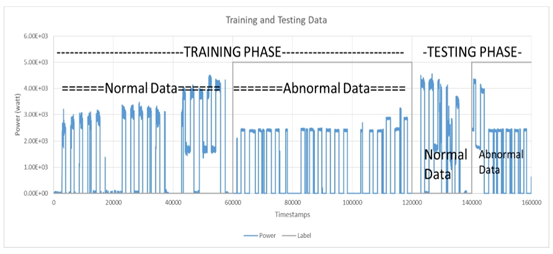
圖二十四、調用Invoke Function

未來考慮將以上的區塊鏈架構擴充至多個區塊鏈[11]。

1. 實驗結果

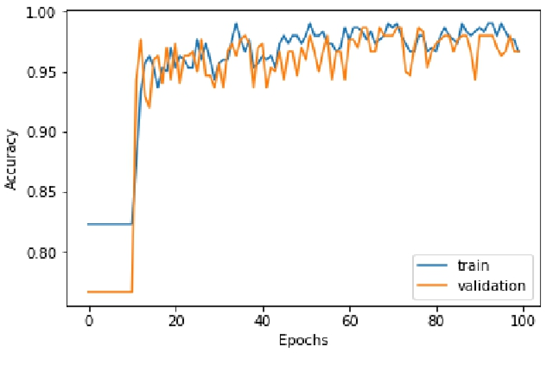
## 計畫第一年，我們已經實作完成智慧閘道器中以上說明的各個部分，並且已經開始整合到Raspberry Pi終點裝置平台。然而，整合的部分工作尚未完成。這部分需要等第二年繼續完成。目前，LSTM異常偵測模型的相關實驗已經完成並且發表於會議中。以下說明此部分的實驗內容。

## 圖二十五顯示了在訓練和測試階段，正常和異常的資料樣本趨勢。

****

圖二十五、在訓練與測試階段的電力資料分析[20]

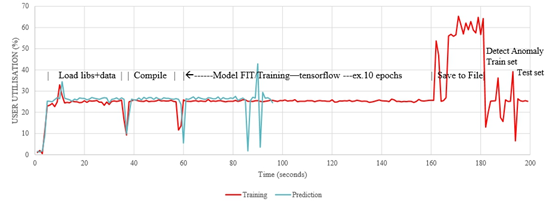
## 圖二十六為訓練model過程中，準確率的趨勢，在圖中可以發現Epochs 超過20後，訓練(Train)和驗證(Validation) 準確度 (Accuracy) 已經達到了93％以上



圖二十六、Accuracy-Epochs分析圖

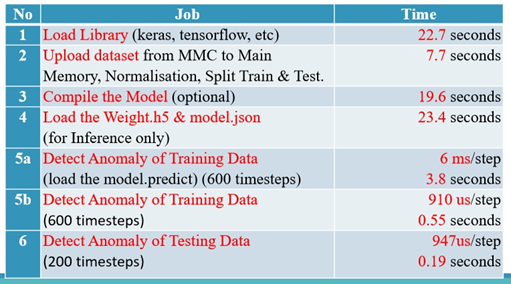
## 測試結果，如圖二十七所示為200個測試數據樣本中，每個樣本由100筆異常和100筆正常樣本組成， 184個樣本被正確預測(TP+TN)，而只有16個預測不正確(FP+FN)，真陽性率（TPR）和假陽性率（FPR）分別為0.81和0.50，其中True Positive = 34，False Positive = 8，True Negative = 8，False Negative = 150，TPR為81%表示34陽性結果(TP)發生在42個陽性樣本(TP+FP)中，準確度(Accuracy)達到了92%。

## 運行時間分析，表二所示，加載函式庫(Library)時消耗22.7秒，正常化的加載資料集消耗7.7秒，編譯(Compile)模型(Model)使用了19.6秒。加載model.json和weights.h5消耗23.4秒。當第一次運行時，由於將動態庫(Dynamic Library)從MMC (Multi-Media Card)加載到主記憶體 (Main Memory)，該程序對600筆樣本進行訓練(Train)消耗3.8秒或大約6 ms/step。連續運行預測，時間減少到0.55秒或大約910 us/step。對200筆樣本進行預測消耗了0.19秒或947us/step同時準確度(Accuracy)也高達92%。



圖二十七、用戶時間分配圖

表二、分配時間圖



1. **結論**

計畫第一年已經設計完成一個基本的AIoT智慧閘道器架構，並且實作到目前大家最常用的Raspberry Pi 3上。論文的部分，目前已經發表一篇國際會議論文 ICCE-TW’2019，報告時也深獲好評。第二年，將整合上述各個模組，並且做整體評估與量測。第三年，我們再將此閘道器應用至其他物聯網領域以及其他智慧技術如推薦系統[12,14]、異質物聯網LoRa網路 [13]、安全機制[15]、基於區塊鏈的自我監控技術[16]、基於區塊鏈與軟體定義(SDN)的分散式霧節點(fog node)架構[17]、基於時空特性的異常偵測技術[19]。

1. 文獻探討:
2. R. Almadhoun, M. Kadadha, M. Alhemeiri, K. Salah. A User Authentication Scheme of IoT Devices using Blockchain-enabled Fog Nodes, In Proceedings of the IEEE/ACS 15th International Conference on Computer Systems and Applications (AICCSA), pp.1-8, 2018.
3. D. B. Araya, K. Grolinger, H. F. ElYamany, M. A. M. Capretz. An Ensemble Learning Framework for Anomaly Detection in Building Energy Consumption. Energy and Buildings, Vol. 144, pp. 191-206, June 2017.
4. A. Diro, N. Chilamkurti. Leveraging LSTM Networks for Attack Detection in Fog-to-Things Communications. IEEE Communications Magazine, Vol. 56, No. 9, pp.124-130, September 2018.
5. A, Dorri, S.S. Kanhere, and R. Jurdak, Blockchain in Internet of Things: Challenges and Solutions, Computer Research Repository (CORR) arXiv:1608.05187v1, http://arxiv.org/abs/1608.05187, August 2016.
6. B. Fekade, T. Maksymyuk, M. Kyryk, and M. Jo, Probabilistic Recovery of Incomplete Sensed Data in IoT, IEEE Internet of Things Journal, Vol. 5, No. 4, pp. 2282-2292, 2018.
7. P. Ferrari, A. Flammini, S. Rinaldi, and M. Rizzi, On the Use of LPWAN for e-Vehicle to Grid Communication, In Proceedings of the AEIT International Annual Conference, pp.1-6, 2017.
8. C. Goursaud, and J. M. Gorce, Dedicated networks for IoT: PHY/MAC state of the art and challenges, EAI endorsed transactions on Internet of Things, 2015.
9. G. Kim, J. Park, and J.Ryou. A Study on Utilization of Blockchain for Electricity Trading in Microgrid. In Proceedings of the IEEE International Conference on Big Data and Smart Computing (BigComp), pp.743-746, 2018.
10. H. C. Lee, and K. H. Ke, Monitoring of Large-Area IoT Sensors Using a LoRa Wireless Mesh Network System: Design and Evaluation, IEEE Transactions on Instrumentation and Measurement, pp.2177-2187, 2018.
11. M. Li, L. Zhu, and X. Lin, Efficient and Privacy-Preserving Carpooling Using Blockchain-Assisted Vehicular Fog Computing, IEEE Internet of Things Journal, Vol. 6, No. 3, pp.4573-4584, June 2019.
12. K. Luo, W. Yu, H. M. Amjad, S. Wang, L. C. Gao, K. Hu. A Multiple Blockchains Architecture on Inter-Blockchain Communication. In Proceedings of the IEEE International Conference on Software Quality, Reliability and Security Companion, pages 139-145, July 2018.
13. R. Mu. A Survey of Recommender Systems Based on Deep Learning. IEEE Access, Vol. 6, pp. 69009-69022, November 2018.
14. M. N. Ochoa, L. Suraty, and A. Duda, Large Scale LoRa Networks: From Homogeneous to Heterogeneous Deployments, In Proceedings of the International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), pp.192-199, 2018.
15. B. Purkaystha, T. Datta, M. S. Islam, M-E-Jannat. Product Recommendation: A Deep Learning Factorization Method Using Separate Learners. In Proceedings of the Product Recommendation: A Deep Learning Factorization Method Using Separate Learners, February 2018.
16. Raspberry Pi 台灣樹莓派. 用Raspberry Pi + LoRa實做微型物聯網閘道器. Retrieved from [http://raspberrypi-tw.s3.amazonaws.com/slideshare/build-a-nano-i -ot-gateway-with-pi-and-lora.pdf](http://raspberrypi-tw.s3.amazonaws.com/slideshare/build-a-nano-iot-gateway-with-pi-and-lora.pdf), Nov 29, 2017.
17. M. Samaniego, and R. Deters, Internet of Smart Things - IoST Using Blockchain and CLIPS to make Things Autonomous, In Proceedings of the 1st IEEE International Conference on Cognitive Computing, 2017, pp.9-16.
18. P. K. Sharma, M.Y. Chen, and J.H Park . A Software Defined Fog Node Based Distributed Blockchain Cloud Architecture for IoT. IEEE Access, Vol. 6, pp. 115-124, 2018.
19. S.-H. Tsang. (2012). Docker Tutorial 3: Running Image. Retrieved from [https://medium.com/@sh.tsang/docker-tutorial-3-running-images-1d -bb76b971af](https://medium.com/@sh.tsang/docker-tutorial-3-running-images-1dbb76b971af) (July 6, 2018).
20. S. Zhou, W. Shen, D. Zeng, M. Fang, Y. Wei, Z. Zhang. Spatial-temporal Convolutional Neural Networks for Anomaly Detection and Localization in Crowded Scenes. Signal Processing: Image Communication, Vol. 47, pp. 358-368, September 2016.
21. D. Utomo, and P. A. Hsiung, Anomaly Detection at the IoT Edge using Deep Learning, In Proceedings of the IEEE International Conference on Consumer Electronics (ICCE-TW), Taiwan, May 2019.