應用元強化學習於雲端應用服務快速在線異常檢測

Applying Meta-Reinforcement Learning to Cloud Application Services for Fast Online Anomaly Detection

報告者:陳冠儒
指導教授: 張保榮 教授

OUTLINE

1. Introduction
2. Related Work
3. Research Method
4. Experimental Results and Discussion
5. Conclusion

1. Introduction

Introduction

- 雲端運算使用虛擬化技術來應付大量的計算需求，成為了現在應用程式提供服務的媒介，使用者可以透過網際網路按需(on-demand)即取的運算。如此一來，資源集成管理減少維護成本，整合了多個異質性系統並在特殊任務上提供強大的運算能力支持。然而，隨著集群的複雜度增加，發生崩潰或停機時總會造成困擾。
- 資料中心有程式監控著數台伺服器的運行狀態，包含處理器、記憶體、磁碟、網路設備等日誌，並在異常狀態發生時發出警告。但雲端服務的動態性導致故障難以預測，許多正常行為不斷的被重新定義，正需要一個主動性的方法來減少故障帶來的影響。

Introduction(cont.)

- 深度學習在許多方面上有很好的成就，但通常需要大量人力生成標籤化資料訓練。在資料會快速變化的環境下，資料分佈發生變化會導致模型失效。
- 時間序列的異常檢測任務是一大挑戰。首先，異常樣本並不常見，導致樣本比例失衡，其次，當發生連續性的異常會導致異常模式難以辨認。在雲端伺服器系統的資源使用率序列中，可以得知使用者行為改變會使得模型無法即時適應。若是在執行高風險應用，發生故障會造成影響。

Introduction(cont.)

- 近年來強化學習演算法的快速發展，很大程度的改善了異常模式難以辨認的問題。深度強化學習可以透過與環境互動學習，並透過經驗調整模型參數形成最優策略。然而，強化學習與環境相依性強，當環境變化時也會導致此前的最優策略無效，對於雲端複雜的環境變化會難以適應。
- 針對此問題，我們將強化學習結合了Meta Learning。Meta Learning演算法致力於少量樣本的快速學習，以過去經驗來對新的、少數的資料做出快速而準確的學習。其中，MAML(Model-Agnostic Meta-Learning)為Meta Learning系列的演算法之一，它的特色在於與模型無關，專注在學習各個任務之間的共通表徵。我們研究此演算法應用於快速適應異常模式的應用。

目的

- 本文研究對象為台灣恩智浦半導體股份公司雲端應用服務，服務包含多項虛擬主機共同運作維持。該公司使用Zabbix Server監測各項服務的數值，並設定超過預設閥值發出警告通知管理者。然而，此方案僅能告知使用率過高的裝置，無法體現整體的故障及實際異常原因。
- 本研究首先分析Zabbix Server監控資料，使用滑動窗口生成時間序列資料以及異常標記，以TSFEL、PyOD套件擷取可轉移的元特徵。再來，我們使用MAML-RL演算法在訓練階段產生不同的子任務進行訓練，記錄策略損失和網路參數。外部的循環我們使用TRPO(Trust Region Policy Optimization)尋找出最優的策略最大化決策的獎勵，以此生成一個初始模型。最後，使用目標資料集進行快速的小步長訓練得到一個在線預測模型。

7

2. Related Work

8

Time Series Anomaly Detection

- 時間序列異常檢測在於辨認一個序列中的潛在異常，然而，時間序列的異常複雜多變，有時會出現一系列複雜的連續時間異常，導致檢測算法難以辨認。因此，時間序列的異常檢測是充滿挑戰的任務，若是發生預測失準將會付出龐大的代價。目前常見方法為:
  1. Traditional statistical methods
  2. Semi-Supervised Learning

9

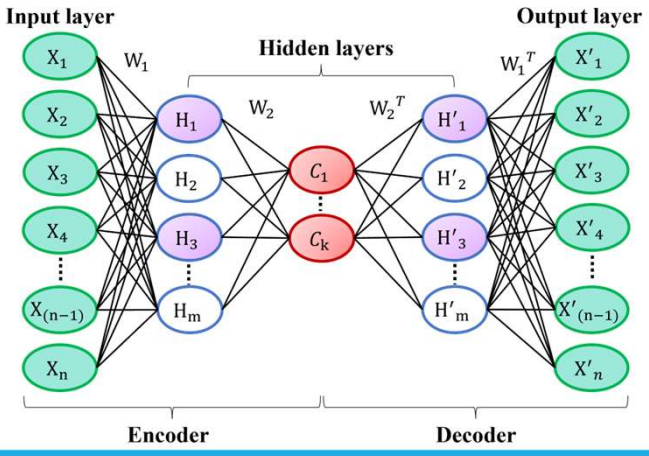
Traditional statistical methods

基於統計的異常檢測是最早使用的方法，它假設了目標資料是常態分佈，當觀測的資料超過三倍標準差之外即判定為異常。此方法簡單且直覺，但當資料不是常態分佈時就會失效。再來，若資料包含高維的資料點，此方法就沒辦法辨認出多為空間的異常。

10

Semi-Supervised Learning

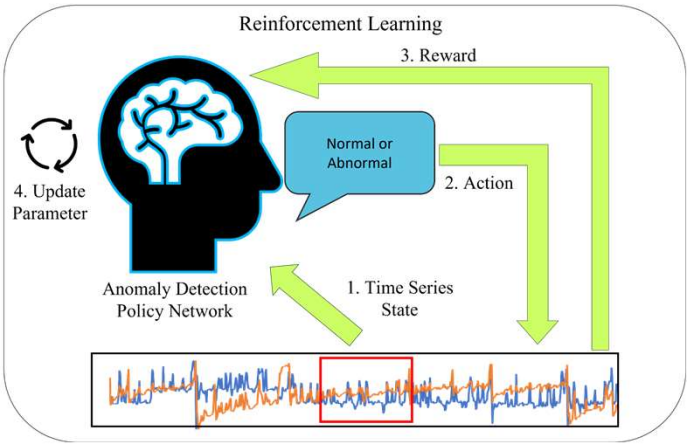
- 基於自編碼器方法包含一個對稱的網路結構以及一個隱含的向量結構，模型會透過還原輸入來學習目標的資料分布，如下圖所示。
- 由於異常檢測時常會受限於陽性樣本稀少的原因，因此自編碼器可以使用模型特性來學習正常資料分布，並判定還原誤差較大者為異常。此方法在某些領域得到極大的成就，但在雲端設施維護的領域上會因為環境快速轉變而失效。



11

Reinforcement Learning

- 強化學習是一種可以透過與環境互動修正代理策略的一種演算法。
- 時間序列的異常檢測將此問題視為馬可夫決策過程(MDP)，其中包含了一個策略網路、回饋以及環境狀態的循環，如右圖所示。代理必須要透過學習控制策略來得到最大化的獎勵。
- 強化學習過度的依賴環境，導致更新模型時需要重新模擬上萬次的過程，提升了困難性。



12

Meta Learning

- 深度學習最大的問題就是對資料量的需求，需要大量的樣本學習去辨認數個物件去模擬人類學習能力，如此一來在資源有限的狀態下很難得到一個準確的模型。
- 元學習是近期迅速發展的一種技術。它希望讓模型擁有「學習如何學習」的能力，由過去經驗中快速地去學習新的任務。
- 人類自出生以來便具備多項優勢，對於一項物件只需要見過幾次便可以分出其差別。因此，相對於機器，「Task」便是此演算法的主要核心。元學習可以透過不同Task的分類經驗學習，並透過先驗知識快速適應新的任務。

Model-Agnostic Meta-Learning(MAML)

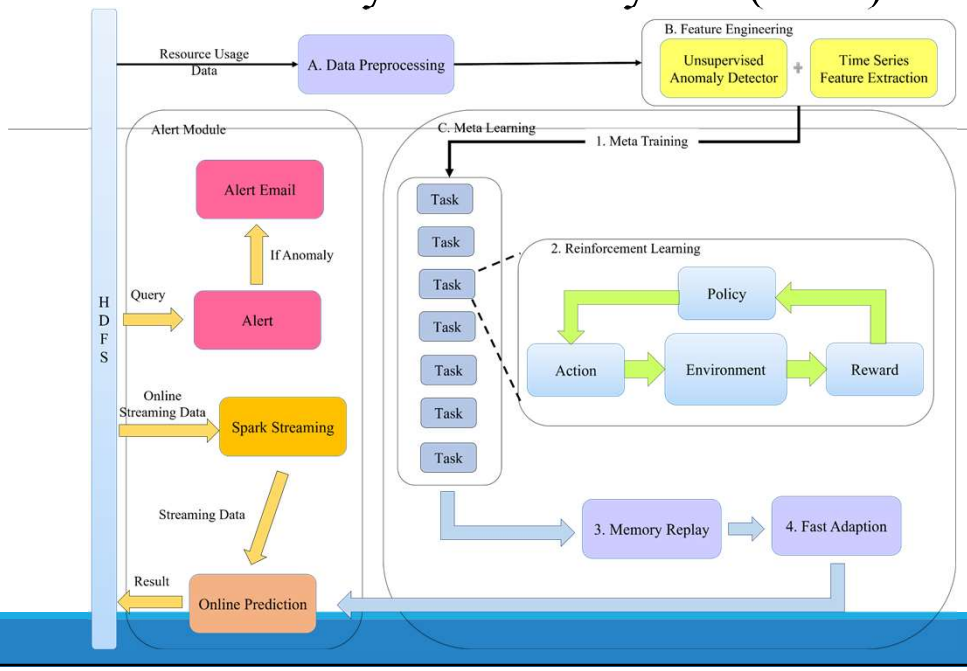
- 在元學習領域中，Model-Agnostic Meta-Learning(MAML)是一個經典的演算法，因為他與模型無關，可以視作一個讓模型自我學習的框架。常常被用來加入其他深度學習模型，甚至是強化學習也能夠無縫嵌入其中。
- MAML會透過多個小型子任務進行分類訓練，並生成一個可以快速適應新任務的初始模型，如Figure 5所示。因為其快速的適應性以及與模型無關的優點，我們將其應用於雲端應用異常檢測系統，除了能夠具備強化學習的自我修正的能力，在雲端環境快速變化時也能夠快速適應。

3. Research Method

Meta-RL anomaly detection system

- 本研究結合台灣恩智浦半導體公司集群建立異常檢測系統，該集群使用Hadoop與Spark框架，資料來源皆是透過HDFS擷取串流資料。本系統首先擷取過去的硬體資源使用率資料進行滑動窗口標記，以PyOD、TSFEL擷取序列的重要特徵後進行Meta-Reinforcement Learning訓練與測試得到初始模型。
- 針對檢測對象我們進行小步長參數更新後佈署至集群。最後，警報系統會隨時詢問模型的預測結果是否有異常，當系統發現有即將到來的故障時會即時通知管理人員處置，減少非預期意外發生

Meta-RL anomaly detection system(cont.)

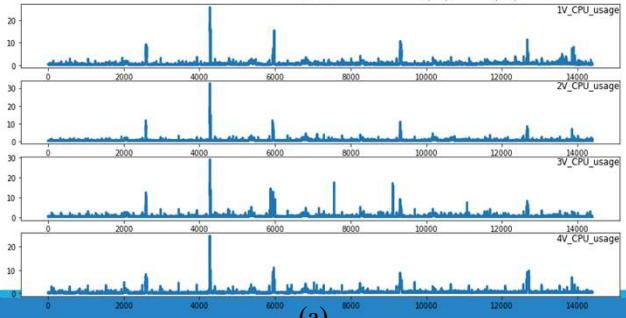


System Abnormal State Analysis

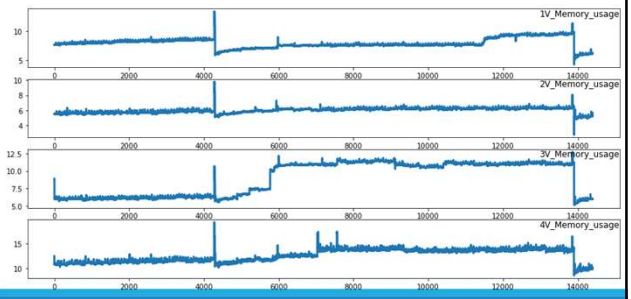
- 本研究對象為台灣恩智浦半導體股份公司用於半導體封測生產線資料的上傳、搜尋與運算的集群，該集群使用Zabbix Server工具監控集群內的各項數值。Zabbix Server是一種伺服器的監控工具，可以監測資料中心內各項服務和資源的使用率，並將監測資料即時寫入CSV檔案儲存。
- 本步驟首先觀察恩智浦公司Oplus應用服務在2021/2/8~2021/3/10之資源使用率資料。我們發現該應用服務的記憶體會在使用率達到高峰時發生一次切換，切換當下 CPU使用率也會達到最高峰，切換後記憶體會維持約20天的上升趨勢，如Figure 7(a)、(b)所示。

System Abnormal State Analysis(cont.)

- 本步驟首先觀察恩智浦公司Oplus應用服務在2021/2/8~2021/3/10之資源使用率資料。我們發現該應用服務的記憶體會在使用率達到高峰時發生一次切換，切換當下 CPU使用率也會達到最高峰，切換後記憶體會維持約20天的上升趨勢，如 (a)、(b)所示。



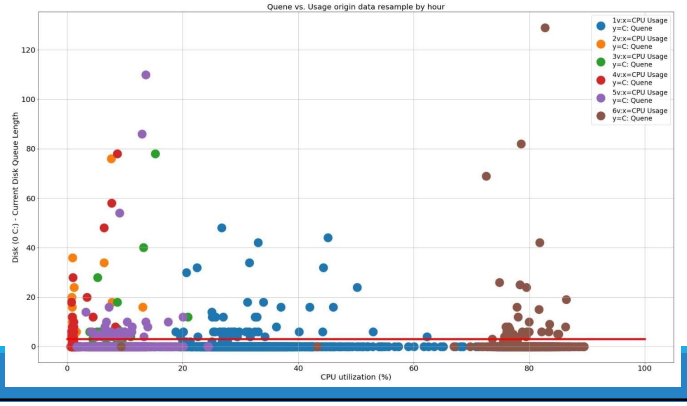
(a)



(b)

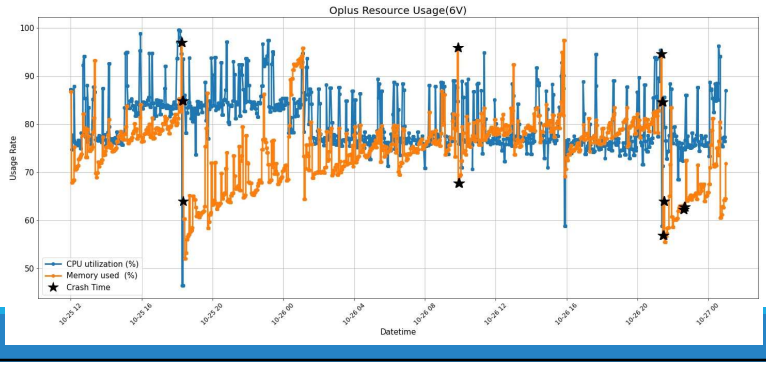
System Abnormal State Analysis (cont.)

- 硬碟佇列(Disk Queue)長度也是一項重要指標，當硬碟佇列長度大於2時代表硬碟此時等待處理的任務過多。當Oplus應用服務的CPU使用率在高使用率與低使用率時有較長的硬碟佇列長度，如下圖所示。



Abnormal Data Labeling

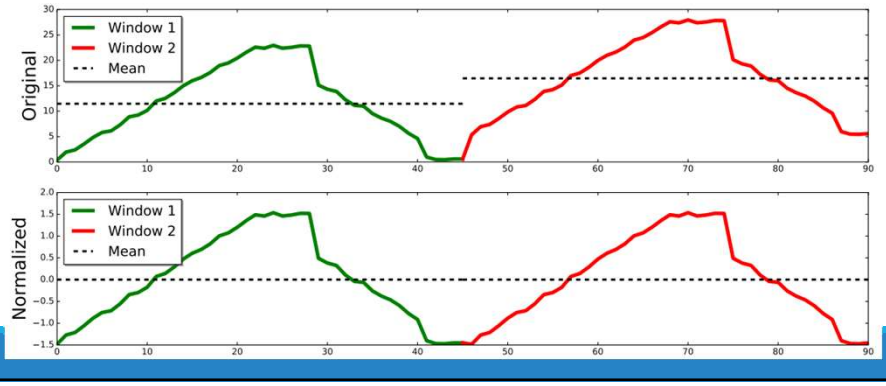
- 資料集為每三分鐘接收一次資料。其中，包含了CPU、Memory、Disk Queue等資料以及不同編號的虛擬機(代號1V、2V、3V、4V.....等)。我們主要觀察系統的CPU與Memory使用率資料作為主要特徵，如圖所示。



21

Abnormal Data Labeling (cont.)

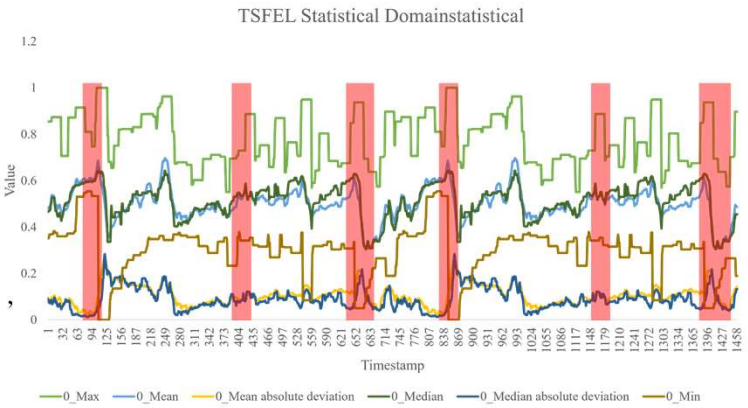
- 對於真實的異常時間是以一固定寬度的窗口標記，滑動窗口內的資料我們採用local normalization化方法，以此方法來顯現出突如其來的資料變化，如圖所示。



22

Meta Feature Extraction

- 本研究針對時間序列做進一步的特徵擷取。首先，原始的時間序列包含了複雜的特徵空間，導致強化學習的訓練效果不佳。因此，我們使用額外工具提取時間序列可轉移的元特徵。
- 我們使用TSFEL(Time Series Feature Extraction Library) 擷取出統計域特徵，如最大值與最小值，梯度等並觀察各項數值與折線圖，如圖所示。



23

Meta Feature Extraction (cont.)

- 由於元學習的過程我們橫跨了不同屬性的資料集，我們需要提取的特徵能夠低度的依賴元資料集，防止當資料轉變時特徵失效。。
- 本步驟使用了PyOD 套件集成OCSVM(OneClass SVM)、iForest(Isolation Forest)以及LODA(Lightweight on-line detector of anomalies)[10]無監督異常檢測模型，使用模型輸出的異常分數作為資料集的狀態映射。
- 我們還加入了樣本與異常資料之間的距離作為特徵。經過元特徵擷取後儲存成CSV檔案作為元學習訓練使用。

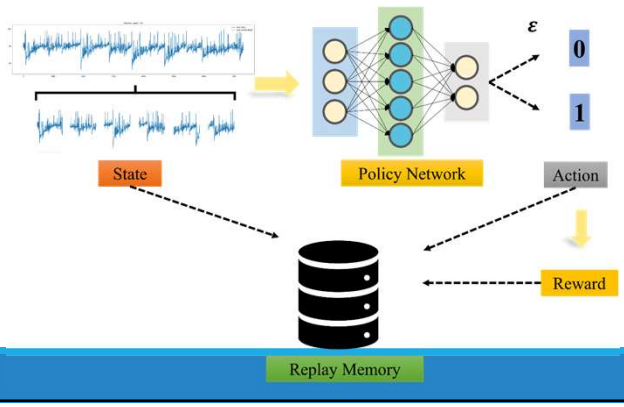
24

Create Meta-reinforcement Learning Tasks

- 本研究使用RLAD方法建立強化學習環境，首先，我們將時間序列使用固定的滑動窗口擷取資料作為環境狀態(State)，並將動作空間(Action Space)限制為兩個離散狀態:0(正常)、1(異常)。
- 將環境狀態作為輸入資料輸入策略網路(Policy Network) 輸出代理人的決策，並且有一個概率ε決定代理是否遵從策略網路提出的建議。

Create Meta-reinforcement Learning Tasks (cont.)

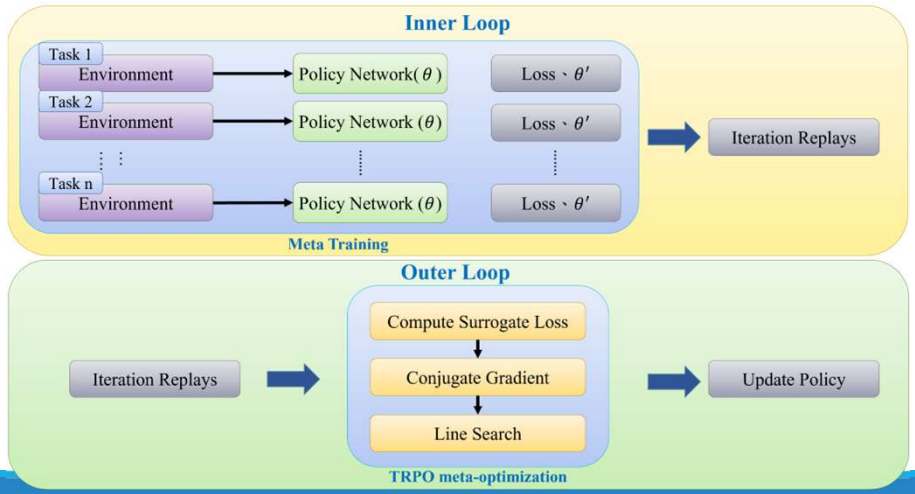
- 輸出的決策會與資料集的標籤計算回饋(Reward)，在此我們設定回饋函式為{TP,TN,FN,FP}。會有一個記憶模組(Replay Memory)儲存此轉移狀態<s,a,r,s'>，如圖所示。



Create MAML Algorithms

- 本研究使用learn2learn套件與python語言開發。首先，我們的Inner Loop建立多個執行緒用於同時在多個強化學習環境下執行，並為每個環境建立一個Task與策略網路(θ)。
- 將每個環境產生的State、Action、Reward、Loss及更新後的策略網路參數(θ^')儲存至Iteration Replay。在Outer Loop部分，我們選擇TRPO(Trust region policy optimization)找出最佳策略。TRPO演算法重複使用策略的Trajectory增加樣本的使用率，也能確保強化學習在訓練的過程不會因為策略的改變影響模型的學習效果，該演算法也透過劃定一個可信任的策略學習區域，保證策略學習的穩定性和有效性。

Create MAML Algorithms(cont.)



Training the Meta Reinforcement Model

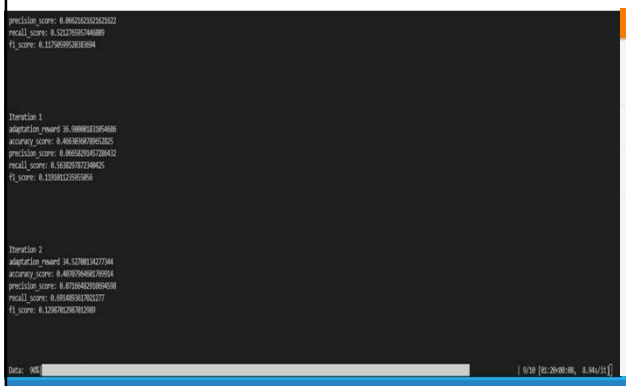
- 本步驟首先使用OpenAI gym套件註冊資源使用率環境作為本實驗的Environment，並設置Inner Loop之超參數如下表所示。

Hyperparameter	Value
Hidden layer	[100,100]
Adapt learning rate	0.5
Number of iterations	100
Meta batch size	10
Number of workers	10
cuda	1

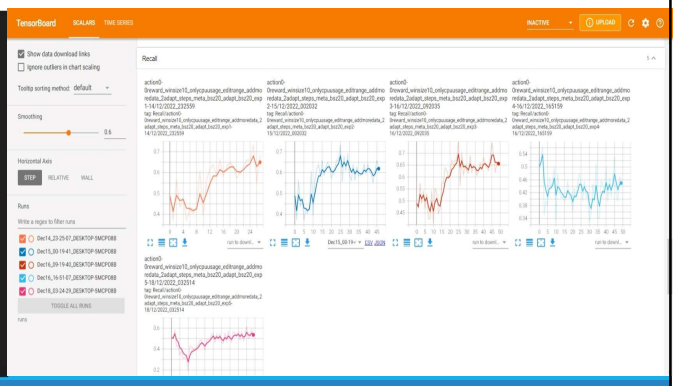
29

Training the Meta Reinforcement Model

- 每個Iteration設置記錄檔紀錄Reward與目前最終模型的準確度，並設置Tensorboard觀察訓練過程趨勢，如(a)、(b)所示。



(a)



(b)

30

Policy Network Evaluation

- 訓練過程我們會觀察每一個迭代累加的Reward數值，並透過觀測趨勢修改環境設定的獎勵與懲罰機制，如圖所示。

31

Training the Meta Reinforcement Model

- 我們使用Precision、Recall、F1-score三項指標評估模型是否達到可以佈署的程度，公式如式(1)、(2)、(3)所示。
- 初始化模型會先以離線的方式接收目的裝置之硬體資源使用率串流資料，經過前處理後進行短步長的梯度更新。最後評估測試損失是否收斂後佈署至目的裝置。

$$Precision = \frac{TP}{TP + FP} \quad (1)$$

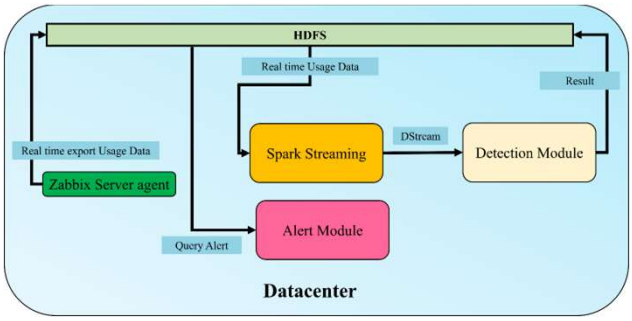
$$Recall = \frac{TP}{TP + FN} \quad (2)$$

$$F1 - score = 2 * \frac{Precision * Recall}{Precision + Recall} \quad (3)$$

32

Deploy the Model to Perform Online Prediction

- 經過評估可使用之模型將會佈署至台灣恩智浦半導體股份公司的運算集群。
- 該公司目前是使用Hadoop與Spark建立集群系統，使用Zabbix Server監控各個虛擬機的資源使用率。系統設定固定時間會將資料匯入HDFS(Hadoop Distributed File System)。
- 本研究使用Spark streaming將資料轉換成DStream(discretized stream)形式輸入在線預測模型進行故障預測，檢測結果將即時儲存至HDFS。再由警報系統發出警報通知管理人員處理，如右圖所示。



33

4. Experimental Results and Discussion

34

Experimental environment

- 本研究使用深度學習工作站進行訓練，配備有Intel® Xeon® Silver 4208和NVIDIA Geforce RTX3080，並有32GB的DDR4記憶體。我們使用Anaconda建立虛擬環境進行開發，使用到的開源軟體版本如下表所示。

Package	Version
conda	22.9.0
Python	3.7.10
Pytorch	1.8.1
CUDA	11.6
cudatoolkit	11.1
cudnn	8.1.0
tensorboard	2.4.1
matplotlib	3.4.2
torchsummary	1.5.1
pyod	1.0.7

35

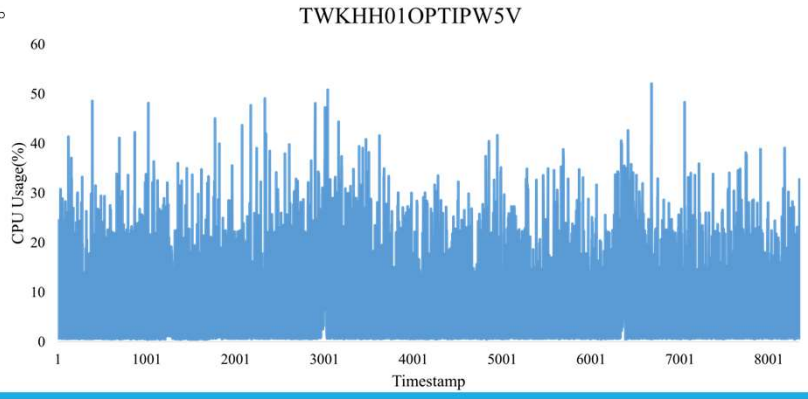
Experimental environment(cont.)

資料集部分使用台灣恩智浦半導體股份公司用於儲存半導體生產線資料的應用服務Oplus，每當機台有新資料產生或工作人員需要詢問資料時會使用。目前時常會發生異常，導致使用者詢問資料時發生延遲。

36

Experimental environment(cont.)

- 我們擷取2021/5~2021/9日期的使用率資料，其中2021/9/1~2021/9/18資料如下圖所示。



37

Experimental design

- 本研究進行了三項實驗。
  1. Experiment 1：使用不同的適應步長來進行訓練，並觀察過程的Reward累積情況。
  2. Experiment 2：計算模型自適應的時間成本比較，觀察不同模型重新訓練的時間成本，評估在線適應的可行性。
  3. Experiment 3：比較我們的方法與現行半監督模型在時間序列的異常檢測效能比較。

38

Experiment 1

- 實驗一測試不同的適應步長對於Reward的影響。我們設定Inner Loop學習率為0.1以及Outer Loop的學習率為0.8，網路結構為兩層長度為100的神經元，並測試適應步長1、3、5下的Reward比較。

39

Experiment 2

- 實驗二計算了模型訓練的時間，分為第一次訓練模型時所花費的秒數以及第二次重新適應新資料花費的時間。
- 第一次訓練時使用的是恩智浦半導體Oplus硬體使用率資料在2021/9/1~2021/11/30期間的使用率共31,920筆資料，作為初始化模型的先驗知識。
- 第二次訓練時使用2021/12/1~2021/12/31期間的資料做為測試資料，時間的計算以模型訓練之F1-score到達0.75的時間。。

Model	First(Ks)	Adaption(Ks)
Autoencoder	0.645	0.0012
Variational Autoencoder	0.815	0.0031
Meta Reinforcement Learning	5.511	0.0003

40

Experiment 3

- 實驗三評估強化學習在雲端服務的時間序列異常檢測的F1-score、Recall以及Precision指數。我們選用目前時間序列異常檢測常用的半監督學習演算法Autoencoder、Variational Autoencoder(VAE)、Temporal Convolutional Autoencoder(TCN-AE)與本文之強化學習演算法比較，如下表所示。

Model	F1-score	Recall	Precision
Autoencoder	0.558	0.6	0.523
Variational Autoencoder	0.445	0.5	0.401
Temporal Convolutional Autoencoder	0.565	0.632	0.511
Meta Reinforcement Learning	0.781	0.772	0.791

41

Experiment 3

- 在異常警報閥值我們採用固定區間內異常窗口數量來判定，若此區間內發生超過閥值數量的異常才判定為異常，以此來防止過度敏感的假警報發生。

42

Discussion

- 在實驗一可以得知在訓練出使模型時可以提高適應步長來訓練，但當適應步長越大時Reward並不會以倍數方式增長，反而會減少增加的幅度。從實驗中發現模型在訓練時大約在第40個迭代數時會逐漸收斂，可以減少迭代次數來加速訓練流程。
- 由實驗二不同的模型適應時間來看，雖然本研究的Meta Reinforcement Learning方法在訓練初始模型時是時間成本最高，但當有新任務產生或是需要更新模型時，進行適應的時間是最少的，可以證明我們的方法可以快速適應變化後的資料集。
- 由實驗三可以得知我們的模型表現優於現今流行的半監督式學習演算法，並且在網路結構上我們僅使用簡單的兩層神經元就可以提升約1.4倍的效能。

5. Conclusion

Conclusion

- 從實驗結果可以看出，本研究減少雲端應用服務異常檢測時，因為使用者行為變化導致模型失效需要重新適應的時間與成本。相較於過去的半監督學習模型，我們提出的方法在準確度方面提升了1.4倍，可以得知此方法在辨認異常的準確率與在線的適應性皆有很好的表現。
- 在本研究中，我們透過加入MAML訓練流程改善模型的準確度與適應性，在少量陽性樣本下我們也得到了很好的成果。本研究提出的方法在雲端應用服務的維護上具備著很好的潛力，但就目前的表現來說，模型還有進步的空間。
- 未來可以透過資料特徵擷取與優化器的設計上嘗試更先進的方法，讓模型的性能可以更進一步。算法也能與實際應用的集群進行更有效率的結合，可以透過調動不同節點的閒置資源加速訓練流程，大程度的發揮本算法的能力。

簡報結束
感謝聆聽