國立臺灣大學電機資訊學院資訊工程學研究所

碩士論文

Department of Computer Science and Information Engineering

College of Electrical Engineering and Computer Science

National Taiwan University

Master Thesis

預測深度學習模型於加速裝置上之執行時間

Predicting the Computation Time of Deep Learning Model on Accelerators

林詠謙

Yung-Chien Lin

指導教授：洪士灝 博士

Advisor: Dr. Shih-Hao Hung

中華民國 108 年 8 月

August, 2019

doi:10.6342/NTU201903950

誌謝

首先在碩士班說長不長說短不短的兩年中，首先要先感謝的是洪士灝教授願意悉心的指導並提供有效的建議以及研究方向，不論是在碩一時接到的產學合作計畫或是在做這份研究時，這兩年皆從洪教授身上學到不少系統相關的知識以及做研究的方式，最後也是這些相關的指導才讓我得以完成這份研究。

另外感謝凌華科技提供我在本研究中所需要用到的機器，若是沒有他們的幫助，這份研究的完整度便會大打折扣，也感謝 PASLab 中的所有同學，在我有研究上有問題時總能提供有效的幫助。

最後要感謝我的父母在這兩年來的支持與金援，還有在我做研究遇到瓶頸時，總會陪伴在我身邊的女友以及妞團的各位好友，因為有你們的支持與鼓勵，我才能順利地面對並度過碩班兩年的種種難關。

doi:10.6342/NTU201903950

中文摘要

隨著深度學習的迅速發展， 而為了提高執行的效率，運作深度學習的硬體也是愈發重要，但效能較高的平臺往往也伴隨著高昂的價格，因此本研究的目標便在於讓使用者能夠快速的推算出一個系統的效能，甚至是可以在取得目標硬體之前便可對其效能做簡易的分析。

目前也有許多相關的研究，但其多是使用公式的方式作效能預測，而此方式往往使用線性的方式做計算，如此便會忽略許多細節。而本研究所用的方式為收集足夠多相關資料，並使用深度學習的方式去學習不同的配置下的執行時間。本研究同時也對不同的神經網路做預測，甚至是未取得的硬體。

關鍵字：深度學習、效能預測、系統效能、執行時間

doi:10.6342/NTU201903950

ABSTRACT

With the rapid development of deep learning, in order to improve the efficiency of implementation, the hardware for deep learning is becoming more and more important, but the platform with higher performance is often accompanied by high prices. Therefore, the goal of this research is to let users can quickly calculate the performance of a system, and even can easily analyze its performance before getting the target hardware.

There are a lot of related researches at present, but most of them use formulas to make performance predictions, and this method often uses linear methods to do calculations, so many details are ignored. The method used in this study is to collect enough relevant data and use deep learning to learn the computation time under different configurations. This study also predicts different neural networks, even for unavailable hardware.

Key Words: Deep learning, Performance prediction, Performance of a system, Computation time

doi:10.6342/NTU201903950

目錄

口試委員會審定書 .....#

誌謝 i

中文摘要 ii

ABSTRACT iii

目錄 iv

圖目錄 vi

表目錄 vii

第一章 研究簡介.....1

1.1 研究動機 1

1.2 相關研究 1

1.3 問題 2

1.4 解決方法 3

第二章 研究背景.....5

2.1 神經網路 5

2.2 影響神經網路效能之軟體參數 6

2.3 神經網路的計算結構及軟體架構 6

2.4 PALEO.....8

第三章 實驗方法.....10

3.1 實作方式 10

3.2 為何使用全連接層？ 11

doi:10.6342/NTU201903950

3.3 模型架構 12
3.4 資料收集 13
3.5 如何收集合適之資料 14
3.6 資料收集與分析工具之自動化 15
第四章 實驗結果 17
4.1 預測單層神經網路 17
4.1.1 未使用批次標準化之結果 18
4.1.2 批次標準化 18
4.1.3 使用批次標準化之結果 19
4.1.3 較高誤差項之觀察 22
4.1.5 TensorFlow Timeline 24
4.2 完整模型之預測 26
4.3 其它硬體架構之預測 28
4.4 未取得硬體之預測 30
第五章 結論與未來目標 35
參考資料 37

doi:10.6342/NTU201903950

圖目錄

圖 1.1 方法架構 4
圖 2.1 神經網路資料流 5
圖 3.1 方法架構 11
圖 3.2 目標模型架構 12
圖 3.3 完整模型之預測 13
圖 3.4 對於不同資料量之訓練曲線 15
圖 4.1 部分訓練資料 19
圖 4.2 不同參數之預測與實際曲線 23
圖 4.3 圖 4.2 之各點相對誤差 23
圖 4.4 TensorFlow Timeline 分析結果 1 24
圖 4.5 TensorFlow Timeline 分析結果 2 25

doi:10.6342/NTU201903950

表目錄

表 1.1 PALEO 之預測結果 3
表 2.1 LeNet 架構 7
表 2.2 AlexNet 架構 7
表 2.3 VGG-16 於不同批次之吞吐量之預測 9
表 3.1 實際自動化之運行時間 15
表 4.1 實驗配置 1 17
表 4.2 各架構之差異 17
表 4.3 未作資料前處理之實驗結果 18
表 4.4 預測全連接層之推論時間 20
表 4.5 預測卷積層之推論時間 20
表 4.6 預測池化層之推論時間 20
表 4.7 預測全連接層之訓練時間 21
表 4.8 預測卷基層之訓練時間 21
表 4.9 預測池化層之訓練時間 21
表 4.10 不同參數之預測與實際之相對誤差 22
表 4.11 AlexNet 26
表 4.12 VGG-16 27
表 4.13 ResNet-50 27
表 4.14 InceptionV3 27
表 4.15 實驗配置 2 28

doi:10.6342/NTU201903950

表 4.16 預測不同硬體架構之卷積層推論時間 29
表 4.17 預測不同硬體架構之神經網路執行時間 30
表 4.18 實驗配置 3 31
表 4.19 預測未取得之硬體之卷積層訓練時間 32
表 4.20 AlexNet (未取得之硬體) 32
表 4.21 VGG-16 (未取得之硬體) 33
表 4.22 ResNet-50 (未取得之硬體) 33
表 4.23 InceptionV3 (未取得之硬體) 33
表 4.24 推論之吞吐量預測 34
表 4.25 訓練之吞吐量預測 34

doi:10.6342/NTU201903950

第一章　　研究簡介

1.1　研究動機

隨著近幾年來深度學習[1]的迅速崛起與發展，讓其在許多過去被視為難題的諸多領域皆獲得巨大的進步及成就，像是在語音辨識、自然語言處理以及電腦視覺中街有許多不同的應用層出不窮，並且在其準確率越來越高的同時，我們也為了讓其應用更有效率而發展出了異質加速平台。隨著深度學習模型的架構越來越多且越來越龐大的同時，異質加速器的發展也越加重要。

然而，這類的加速平台通常皆伴隨著高昂的價格，因此在挑選所需要的機器時便已是一個重要的課題，雖然目前也可以租用雲端硬體以提供深度學習的應用，但是一個深度學習模型的訓練通常會耗費好幾個小時的時間，如此一來，使用雲端服務也是會造成相當高的時間及金錢成本。再者，將資料上傳至雲端也難免會有機密資料外洩的疑慮。

而且，即便深度學習領域的專家也很難為其應用挑選出一個合適的加速器並且在實際執行之前便分析出該應用或者是該加速器的效能。因此，這項研究即是讓使用者在實際訓練深度學習模型或是取得加速器之前，便可事先得知其模型以及加速器之效能。

1.2　相關研究

類似的研究像是 DAWNNbench[4]以及 MLPerf[5]他們兩者皆是以 benchmark 的方式做預測，因此，它們僅僅只能針對現有的模型架構亦或是類似的架構作預測，

doi:10.6342/NTU201903950

若是出現了全新的架構則兩者所做出的預測則會相當不準確。

另外有一個名為 PALEO[2][3]的開源工具，其是利用公式的方式去計算並且預測執行時間，但是在進行預測之前，我們必須要計算出一個名為 Platform Percent of Peak (PPP) 的參數，這項參數所代表得意義為：今天一個深度學習模型在執行時，其硬體在執行的時段中，有多少的時間比例是使用其峰值性能在做運算。如果使用者無法得到該參數，則無法透過 PALEO 得到足夠準確的預測。

1.3 問題

上述的幾種方式不論是針對深度學習模型的訓練亦或是推論，皆無法準確地預測執行時間，尤其是全新的深度學習模型被提出或者是供應商發表了全新的硬體架構時。

這些方法能得到的結論僅僅局限在當“深度學習模型變得更深且更龐大或是其輸入參數更多樣時，模型的執行時間會得到相對應的增加，但具體的比例或是提升的效能部分皆無法提供給使用者。以前一章節提到之 PALEO 為例，經過我們實際使用後，下表 1.1 為其對不同卷積神經網路在推論時之執行時間的預測值與實際值之比較，其測試平台為 Nvidia GeForce GTX 1080 且批次的大小為 32。

可以觀察到除了 VGG-16 之外的卷積神經網路之預測時間皆與實際執行時間相差許多，而造成 VGG-16 能有如此高之準確率也與其架構有關係。其中這部分我們將在下一章做更詳細的說明。

doi:10.6342/NTU201903950

表 1.1　PALEO 之預測結果

	Prediction (ms/batch)	Actual (ms/batch)
AlexNet	5.29	9.21
VGG-16	111.6	112.52
ResNet-50	25.1	70.6
InceptionV3	41.3	96.24

1.4　解決方法

由上一章節即可看出公式法並無法準確的預估卷積神經網路之執行時間，因此我們提出了另一個方式來做深度學習的效能預測，我們所使用的方式為建造一個深度學習模型，並且將相對應的軟硬體參數餵進這個模型中，讓其自行學習各個參數之間的差異並且計算出正確的預測。

使用這樣的方式，我們便只需要收集不同參數所對應的執行時間並將這些資料通通丟給深度學習模型即可。如此一來，我們便不需要自己去做公式推導的工作亦可不需要完全地去了解底層硬體是如何去作資料的傳輸及運算。

下圖 1.1 便是我們所提出的方法架構，而其中各個階段的細節處理則會在第

doi:10.6342/NTU201903950

四章作更詳細的說明。

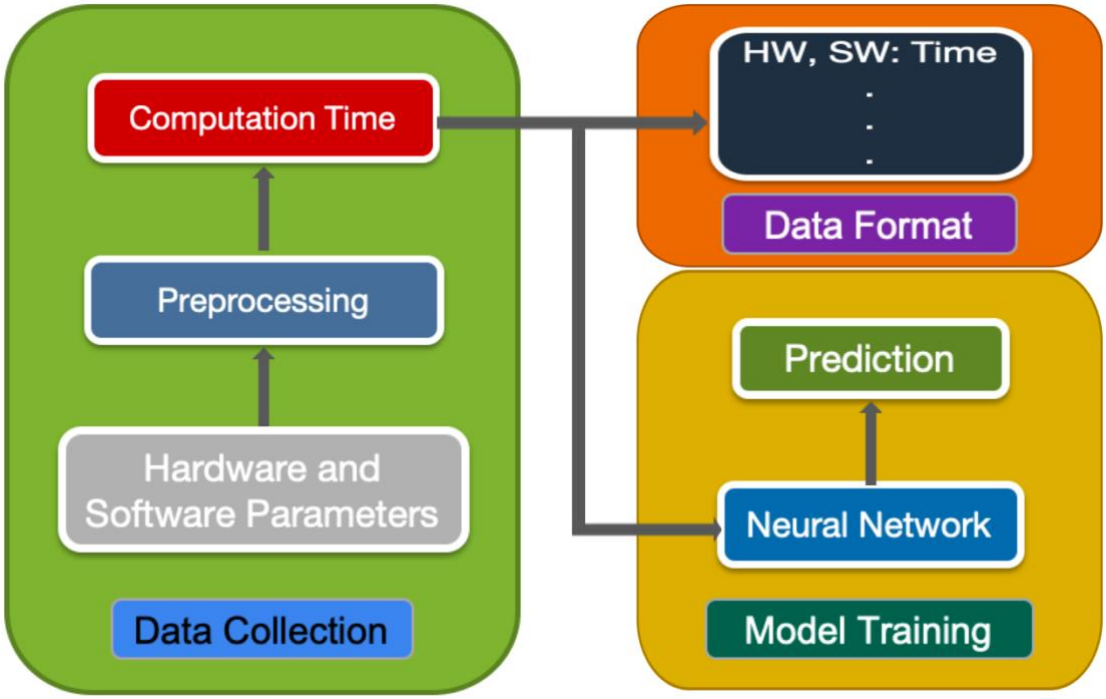


圖 1.1　方法架構

doi:10.6342/NTU201903950

第二章　　研究背景

2.1　神經網路

在這裡先介紹神經網路所做的運算，在神經網路中主要分為兩個階段：訓練以及推論。在訓練的階段中，輸入的參數在經過神經網路的運算並得到結果後，會再與正確答案做比對，並藉由比對的結果再去更新各層的權重及參數。而在推論時，輸入的參數則是輸入已訓練完成之神經網路並由其產生預測結果。

下圖 2.1 則更具體地顯示了訓練及推論兩者之間的差異，由圖中可看出，不論是訓練或推論的過程中，兩者皆由相同的起點進行前向傳播，並經由神經網路計算結果，推論在這時即結束所有過程；而訓練則會繼續，透過與正確答案做比較之後，神經網路便會利用反向傳播去更新各層中各個神經元之權重以及參數，並使得準確率可以更加提升。另外由圖中也可知為了提高效率，訓練的過程中每一批次所含有的資料量會較推論為高。

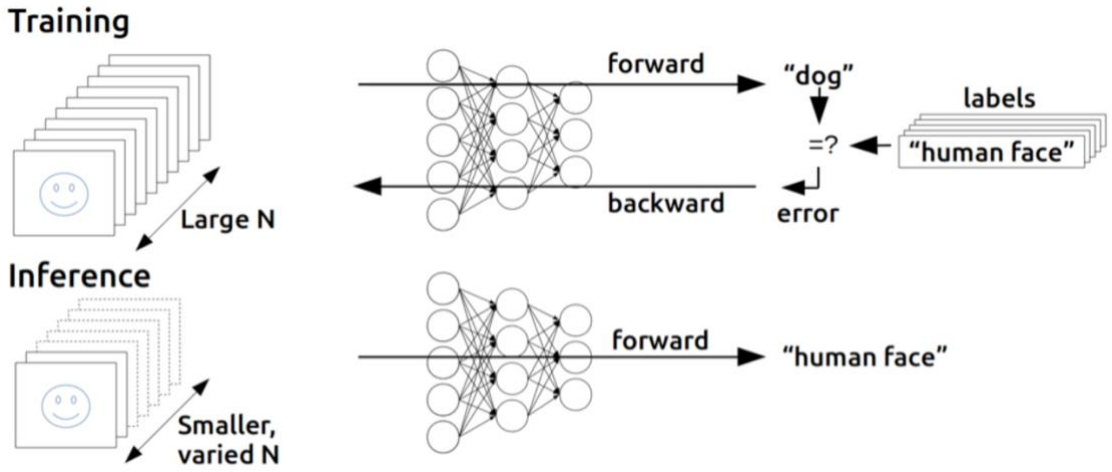


圖 2.1　神經網路資料流

doi:10.6342/NTU201903950

2.2 影響神經網路效能之軟體參數

由上一章之圖 1.1 可知我們需要收集軟體參數，所謂軟體參數即為該神經網路層之架構，其中包含了圖片大小、批次大小、計算核心大小、維度、步長以及神經元個數，而通常便是這些軟體參數會影響到各層之效能差異。

其中圖片大小、批次大小以及維度三者共同決定了輸入資料的大小，其中當然數字越大將會執行更久的時間；計算核心的大小為每一次的運算中包含了多少個數字，因此其數字越大，執行時間亦越久；步長即為核心每次計算完後移動的長度，亦可想像為圖片壓縮的倍率，若步長為 1 則不壓縮；步長為 2 則會將圖片壓縮為四分之一倍，以此類推，步長越大執行時間則越短；神經元個數為該層中包含了幾個神經元，若是越多則執行時間亦越大。

2.3 神經網路的計算結構及軟體架構

除了軟體參數會影響神經網路的效能外，神經網路的長相及運行神經網路的硬體亦對執行效能有相當大程度的影響。神經網路的長相不外乎就是其組成架構或是深度，例如以 LeNet 以及 AlexNet 相比，下表 2.1 與表 2.2 即為兩者之架構差異，最為明顯的可以看到除了層數不一樣外，其中各層的參數以及層與層之間的組合亦不相同。這樣的不同除了造成兩者執行時間上的差異之外，兩者在準確率上亦有差異。

doi:10.6342/NTU201903950

表 2.1　LeNet 架構

Layer		Feature Map	Size	Kernel Size	Stride	Activation
Input	Image	1	32x32	-	-	-
1	Convolution	6	28x28	5x5	1	tanh
2	Average Pooling	6	14x14	2x2	2	tanh
3	Convolution	16	10x10	5x5	1	tanh
4	Average Pooling	16	5x5	2x2	2	tanh
5	Convolution	120	1x1	5x5	1	tanh
6	FC	-	84	-	-	tanh
Output	FC	-	10	-	-	softmax

表 2.2　AlexNet 架構

Layer		Feature Map	Size	Kernel Size	Stride	Activation
Input	Image	1	227x227x3	-	-	-
1	Convolution	96	55 x 55 x 96	11x11	4	relu
	Max Pooling	96	27 x 27 x 96	3x3	2	relu
2	Convolution	256	27 x 27 x 256	5x5	1	relu
	Max Pooling	256	13 x 13 x 256	3x3	2	relu
3	Convolution	384	13 x 13 x 384	3x3	1	relu
4	Convolution	384	13 x 13 x 384	3x3	1	relu
5	Convolution	256	13 x 13 x 256	3x3	1	relu
	Max Pooling	256	6 x 6 x 256	3x3	2	relu
6	FC	-	9216	-	-	relu
7	FC	-	4096	-	-	relu
8	FC	-	4096	-	-	relu
Output	FC	-	1000	-	-	Softmax

而硬體又是如何影響神經網路之效能？我們知道現今最常使用來加速神經網路的平台即為 GPU，會使用 GPU 最主要原因為其平行化的運算架構剛好與運行神經網路時所需要的大量矩陣運算相匹配，再加上 CUDA 架構的興起，使得在 GPU 上編譯程式更為容易。而 GPU 最主要影響深度學習效能的參數即為 CUDA 核心數目，越多的核心即可做越多的平行化運算，便可藉此提供更優異的效能。另外，Nvidia 最新發布的幾個架構更加入了 Tensor 核心，即是專門對深度學習所設計的，

doi:10.6342/NTU201903950

該核心主要增加了混合精度的運算效能。

而影響深度學習效能的硬體參數當然不只有上述的兩種核心，其他像是 GPU 之記憶體大小以決定每次可容許的計算量；記憶體帶寬以決定資料的傳送速度；時脈頻率以決定計算的速度等等不同的因素。甚至 CPU 傳送資料的速度亦會影響，但本研究主要是針對 GPU 上之執行時間做預測，因此 CPU 之影響因素暫不在本文中討論。

2.4 PALEO

由前一章之表 1.1 的結果可觀察到僅有 VGG-16 之預測準確率較高，而造成 VGG-16 能有如此高之準確率也與其架構有關係。VGG-16 總共有 16 層，而其卷積網路層便佔有 12 層之多，再加上卷積神經網路層之計算量較高，因此造成加速器使用峰值效能運作之時間隨之上升，因此在這樣的參數下才導致 VGG-16 之準確率高於其他卷積神經網路。

而我們也針對 PALEO 對 VGG-16 於其他大小之批次是否有一樣準確的預測做了實驗，下表 2.3 便以吞吐量之方式表示其預測結果，可以看到 PALEO 之預測值皆相同是因為在其公式之中認為批次的大小與執行時間會呈線性關係，但實際上以吞吐量來看的話，隨著批次大小的增加，吞吐量應要有相對應的上升以提高效率。

doi:10.6342/NTU201903950

表 2.3　VGG-16 於不同批次之吞吐量之預測

	Prediction (image/sec)	Actual (image/sec)
1	286.53	102.3
2	286.53	145.41
4	286.74	188.54
8	286.74	216.68
16	286.74	268.27
32	286.74	284.39

由上述兩個實驗可知，雖然在批次大小為 32 之時，PALEO 在預測 VGG-16 之執行時間有很高的準確率，但也僅僅只是因為其執行時的所需的效能相當於機器的峰值效能，而在不同的批次大小時，也不能僅僅只用一個線性的公式去做代入。

doi:10.6342/NTU201903950

第三章　　實驗方法

相較於前章所提到的 PALEO 使用公式的方法，我們所使用的方式為利用深度學習的方式去做預測。由 PALEO 的公式可以得知，不同種類的神經網路在不同的參數下，皆會影響其執行時間。因此，若是能收集足夠多的參數之排列組合與相對應的執行時間的話，便可將這些資料當成訓練資料輸入至我們所建造之全連接神經網路之中，並且由該神經網路幫助我們進行效能預測。

而上述提到的參數包含了有軟體及硬體的參數，其中軟體參數包含了輸入影像的大小、批次的大小、核心的大小、維度、步長以及神經元個數；而硬體的參數則是包含硬體架構、CUDA 核心數目、峰值效能、記憶體帶寬以及時脈頻率。

3.1　實作方式

在此我們將詳細說明方法架構中各個階段的細節處理，下圖 3.1 為第一章所展示之方法架構圖，圖中可以看到主要可以分成兩個部分：一是資料的收集，再來則是深度學習模型的訓練。

在資料收集的步驟中，我們輸入了所以可能的軟硬體參數，並且將相對應的時間記錄下來最終以 csv 文檔的方式保存，並且講其當作是日後的訓練資料。而在資料收集完成之後，便到了訓練的階段，而在將資料輸入至神經網路之前，我們會先對其進行簡單的前處理，首先將所有資料以 numpy array 的形式讀取，並且將其切割成輸入參數以及執行時間，有了這樣的前處裡後，我們便可開始進行深度學習模型之訓練階段。

doi:10.6342/NTU201903950

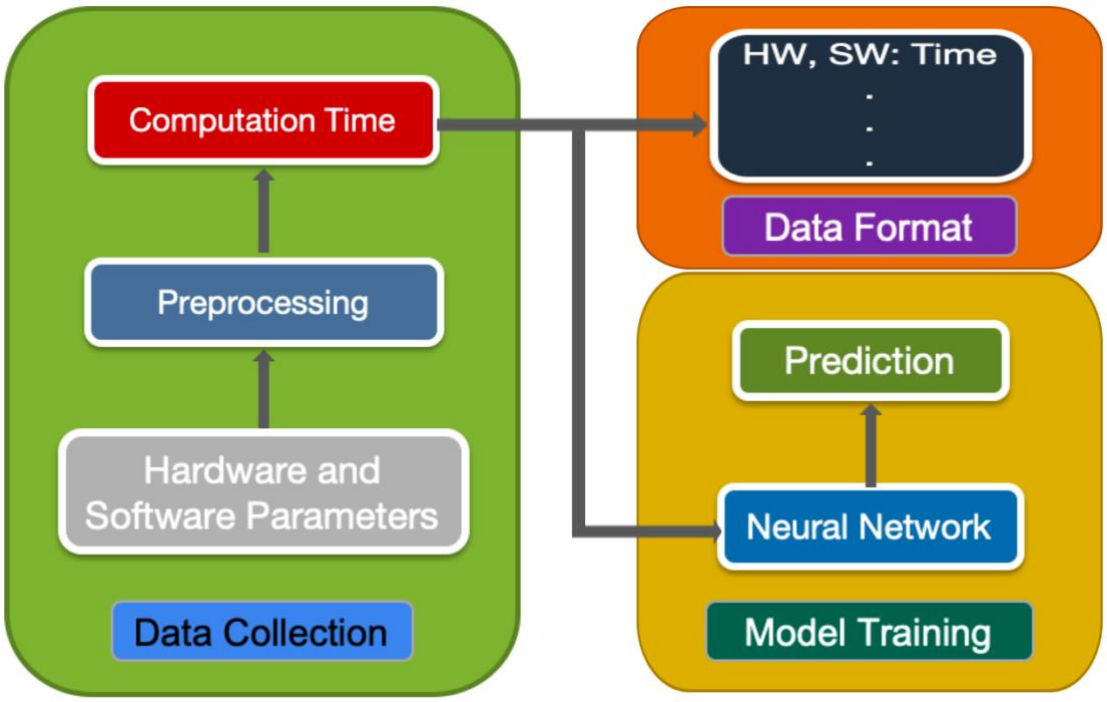


圖 3.1　方法架構

而這樣的方式下，我們所做的預測皆是以單層的神經網路為主，若是要預測一整個神經網路模型的話則如 (3.1) 所示，其中 T_b 表示該神經網路執行一個批次知時間，而 l 則表示不同種或不同參數的神經網路層 $T(l)$ 則為該層之執行時間，因此整個神經網路層之執行時間即為不同層之加總。

$$T_b = \sum T(l) \tag{3.1}$$

3.2　為何使用全連接層？

本章節將說明為什麼我們會選用全連接層來建造我們的神經網路模型。在目前的深度學習應用當中有四種主流的神經網路層，其中包含：卷積層[6]、池化層、長短期記憶層[7]以及全連接層。其中每一種都有自己所擅長的工作性質，像是卷基層擅長影像辨識；池化層主要用於保留影像特徵並同時做影像的壓縮；長短期記憶層則常用於自然語言的處理；最後的全連接層則視為分類器的角色。

doi:10.6342/NTU201903950

從上述的結果可以知道，由於此研究所輸入的參數包含了許多不一樣的數字，並且需要藉由這些參數的不同去計算出相對應的結果，如此一來，全連接層便是最好的選擇，因為全連接層可以幫助我們處理許多不同的參數並藉由學習的方式整理並推導出計算公式，這也是我們選用全連接層的主因。

3.3 模型架構

經由前兩章節的說明後，這邊我們整理過後以圖示的方式來呈現我們所要實作的神經網路模型，如下圖 3.2 所呈現。

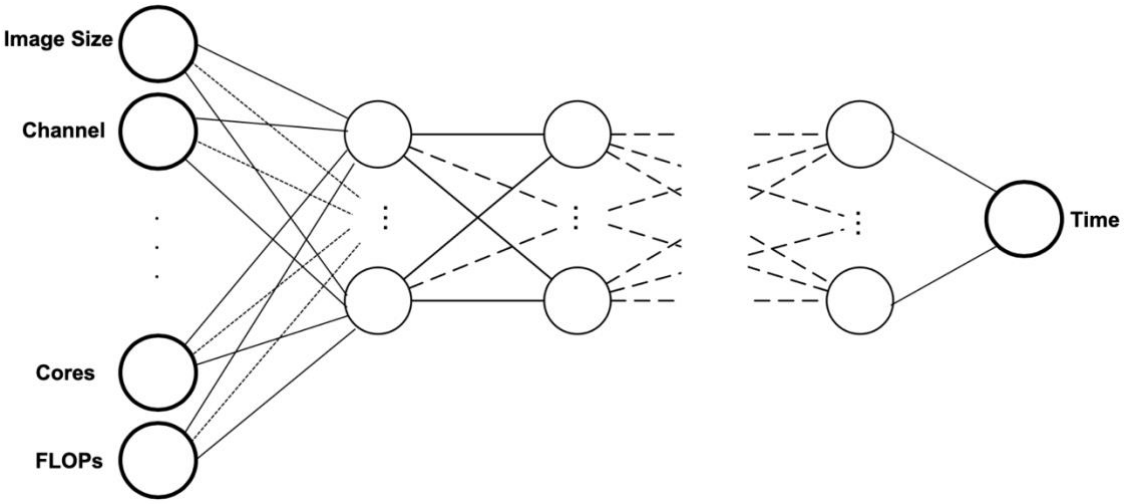


圖 3.2 目標模型架構

由上圖可看到所有的軟硬體參數在一開始輸入至神經網路模型後，變經由中間各層的計算最終得到的結果便為所要預測之神經網路層之執行時間。

doi:10.6342/NTU201903950

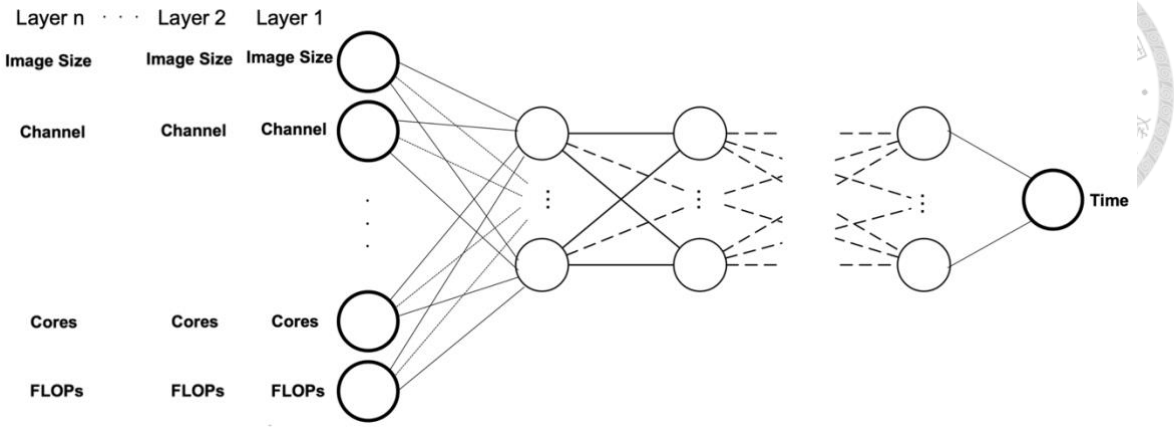


圖 3.3　完整模型之預測

上圖 3.3 為預測完整模型之方式，首先由解析器將神經網路各層之參數配置讀出，爾後將資料一一送入預測模型之中，得出各層之執行時間後，再由 3.1 章節中之公式 (3.1) 作加總。

3.4　資料收集

本節將說明我們如何挑選出足夠也有有的資料來當作我們的訓練資料及。首先，我們所使用的是 TensorFlow[8]的 API 包含有: tf.layers.dense, tf.layers.conv2d 以及 tf.layers.pooling2d 使用這三個函數，對它們輸入不同的參數並測量且紀錄其執行時間。

而可輸入的軟體參數每一項的範圍皆不盡相同，像是影像的大小為 1*1 至 256*256，批次大小為 1 至 64…諸如此類，又每一種神經網路層可輸入之參數也不相同，若是參數一個一個輸入並計算的話，dense 將有 68,719,476,736 種可能；conv2d 將有 3,856,746,414,080 種可能；pooling2d 則有 738,197,504 種可能，以 conv2d 為例，即便每一種可能接只要算 1 毫秒，這樣的排列組合算完也需要 120 年的時間。

由上述的例子可知，即便我們想收集一個完整的資料集並使用最簡易的查表

doi:10.6342/NTU201903950

的方式來做效能預測也是沒有辦法的。因此我們現在所需要的 即為足夠多的資料集，並利用此資料集來訓練出足夠強大的深度學習模型，而在這眾多的排列組合中，我們從 dense 挑選出將近 10,000 種；從 conv2d 中挑選近 55,000 種；從 pooling2d 中挑選近 800 種排列組合，接下來，我們將說明我們是如何挑選並為何認為這樣大小的資料量是足夠的。

3.5 如何收集合適之資料

本章節將討論我們如何去選擇合適的訓練資料，首先由影像的大小開始，主要以 2 的冪次方為主，但太小的直接剔除，因此最小的影像為 5*5 一直到 256*256，另外再加上觀察一些曾經取得不小成就的卷積神經網路像是 LeNet[9], AlexNet[10], VGG-16[11], ResNet 以及 InceptionV3 的架構以及目前最主流的測資像是 ImageNet[12]後，再加入幾項像是 28*28, 112*112, 224*224 等較特別的圖片尺寸；而批次的大小則都為二的冪次方，核心的大小則是選用 1*1 至 11*11 之所有奇數，而剩下的參數像是維度、步長以及神經元個數則是以上述之卷積神經網路之架構為主。使用這樣的方式便可有效的過濾不必要的參數並且也符合目前主流之參數配置。

而資料量大小是否足夠的部分，我們已訓練曲線來做呈現，如下圖 3.4 所示，橫軸所表示的是不同的資料量大小，而縱軸則是相對應的平均誤差，其中誤差如 (3.2) 所表示，是以相對誤差的方式來做計算，在本研究中所有的誤差皆以相同的方式去計算。我們所選用的資料量大小由 50 開始以每次兩倍的方式地增至接近 55,000，可以看到隨著資料量的增加，誤差有下降的趨勢，一直到資料量大約為 20,000 時便開始趨緩，而此時的平均誤差大約都落在 5%至 8%，但由於收集 55,000

doi:10.6342/NTU201903950

筆資料的時間相較於 20,000 筆的時間上只增加了大約五分鐘，因此我們選擇較大的資料量來做訓練。

$$\text{Error} = \frac{\text{Abs(Prediction-Actual)}}{\text{Actual}} \tag{3.2}$$

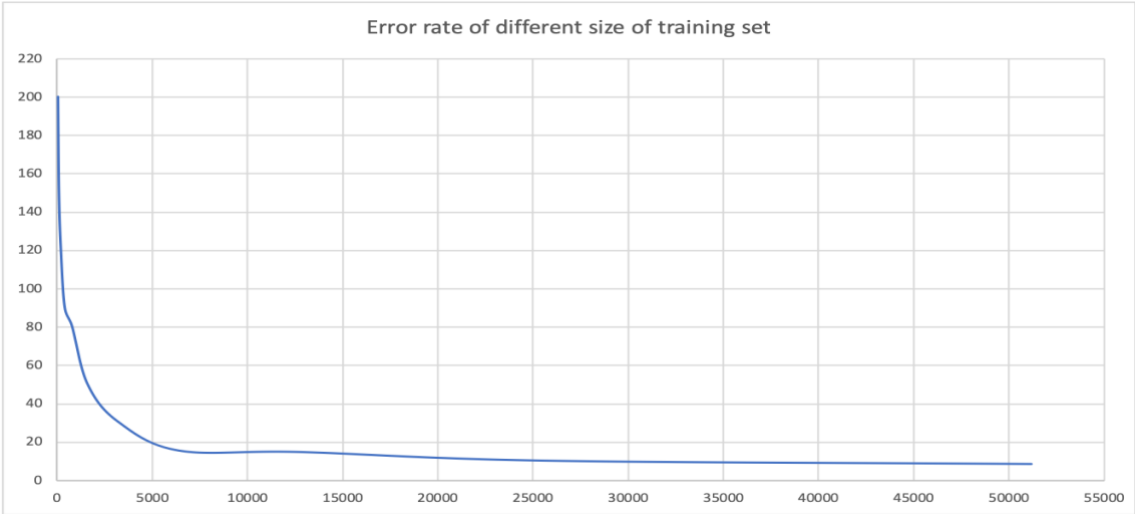


圖 3.4　對於不同資料量之訓練曲線

3.6　資料收集與分析工具之自動化

而本研究所做的工具是可以以全自動化的方式去實現並完成資料收集以及訓練模型兩步驟。而表 3.1 為實際執行之結果，選用的平台為 Nvidia GeForce RTX 2080 Ti，首先在資料收集的部分，對於所有可能的軟硬體參數的計算時間大約會花上半個小時的時間，而在之後的訓練模型的步驟中大概會花上兩個小時左右，因此，在三個小時以內便可以完成一個訓練好的深度學習效能預測模型。

表 3.1　實際自動化之運行時間

Actual Execution Time of Data Collection & Model Training	
Data Collection	**Model Training**
25.6min	**2hour 13min**

doi:10.6342/NTU201903950

而藉由參數的調整，即便今天供應商在上午發表了一部新的加速器，我們也可以使用相同的方式，讓我們在取得實際的機器之前便可以對新的機器作簡單的效能分析。

doi:10.6342/NTU201903950

第四章　　實驗結果

4.1　預測單層神經網路

下表 4.1 為本次實驗所使用的機器，本研究主要皆以 GPU 為主，而在表中可看出四台機器包含了三種不同的架構：Pascal，Volta 以及 Turing，而各架構之間的差異則如下表 4.2 所表示。

表 4.1　實驗配置 1

	Hardware	OS	Software
Node 1-1	Intel(R) Xeon(R) CPU E5-2620 v4 @ 2.10GHz Nvidia GeForce GTX 1080	Ubuntu 18.04	Tensorflow-GPU 1.13
Node 1-2	Intel(R) Core(TM) i9-9960X CPU @ 3.10GHz Nvidia GeForce RTX 2080 Ti	Ubuntu 18.04	
Node 1-3	Intel(R) Xeon(R) CPU E5-2623 v4 @ 2.60GHz Nvidia Titan V	Ubuntu 16.04	
Node 1-4	Intel(R) Xeon(R) CPU E5-2620 v4 @ 2.10GHz Nvidia GeForce GTX 1070	Ubuntu 18.04	

表 4.2　各架構之差異

	Pascal	Volta	Turing
Release Date	5.6.2016	6.21.2017	9.20.2018
Process	14/16nm	12nm	12nm
Core	Cuda only	Cuda and Tensor	Cuda, Tensor, and RT
Application	Gaming, Workstation	AI, Workstation, Datscenter	AI, Workstation, Gaming
Device	Node1-1, Node1-4	Node1-3	Node1-2

由上表可知發布日期可看出迭代的順序為 Pascal、Volta、Turing，而 Pascal 上一代的產品為 Maxwell 但本研究無使用該架構，因此沒有列出。而三者最主要的

doi:10.6342/NTU201903950

差異在於核心，三者皆含有 Cuda 核心，此核心主要用於平行運算，而後兩者則又在此基礎上加上了 Tensor 之核心，便是這樣的差異，使得後兩者在做人工智慧或是深度學習的應用時會得到更佳的效能。另外 Turing 相較於 Volta 則又多了 RT 核心，此核心的工作主要在於做即時的光線追蹤，因此在遊玩遊戲上相較其他兩者會有更佳的體驗，但本研究不會運用到此核心，因此這裡也不作過多的說明。

4.1.1 未使用批次標準化之結果

若是我們不對所收集的資料做前處理的話，所訓練出來的模型，其準確率將會非常差，如下表 4.3 所表示，不僅僅最大誤差與平均誤差都相當大，就連標準差亦是如此，這表示的不僅僅為數值分佈鬆散，也表示模型可能出現了過擬合的現象，即是模型對某些範圍之數值預測相當準確，而其他則誤差非常高。

表 4.3 未作資料前處理之實驗結果

Predict Inference Time on Node 1-1			
	Max	Standard Deviation	Average
Fully Connected	89.8%	30.1%	42.8%
Convolutional	113.4%	35.9%	50.1%
Pooling	77.2%	26.1%	38.8%

4.1.2 批次標準化

由於前段所展現的結果相當不理想，因此我們開始從所收集的資料來檢查是否其中有什麼錯誤，下圖 4.1 為訓練資料之部分截圖，各個參數由左到右分別為影像大小、維度、核心大小、批次大小、神經元個數、步長以及最後是前幾項參數所

doi:10.6342/NTU201903950

對應之實際執行時間。

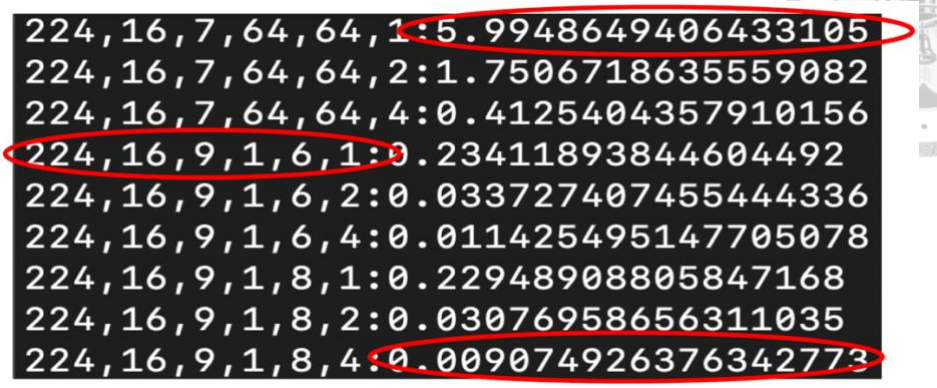


圖 4.1　部分訓練資料

由上圖紅色圈所標示的位置可以看出，在所輸入的資料中，其中各參數之間的範圍相當廣，尤其是在最後的執行時間部分，有些差距甚至會到達 1,000 倍以上，而由於大部分的執行時間皆較短，再加上有如此巨大的差異才會導致預測的不準確，因此我們需要方法以縮小數值的範圍。

而批次標準化便是一個相當有效且容易的實作方式，其主要的功能便是將輸入資料的範圍縮小至 0 到 1 之間，而 TensorFlow 剛好也有提供 API 供我們使用，但由於通過 TensorFlow 之標準化函數之後之數值無法還原，因此我們只針對軟硬體參數的部分使用 TensorFlow 之方式，對於執行時間的標準化我們則是利用同乘一個常數後再開根號的方式去實作標準化，藉此我們才有辦法將其數值還原成實際之執行時間。

4.1.3　使用批次標準化之結果

下表 4.4 至表 4.9 預測池化層之訓練時間則為資料經過批次標準化之後之預測結果，可以看到各項誤差皆有所下降，並且平均也都有得到九成左右之準確率。

doi:10.6342/NTU201903950

表 4.4　預測全連接層之推論時間

Predict Inference Time				
	Max	Standard Deviation	Average	
Fully Connected	16.2%	5.2%	8.6%	Node 1-1
	19.5%	4.3%	4.3%	Node 1-2
	19%	5.1%	8.2%	Node 1-3
	20.2%	6%	7.3%	Node 1-4

表 4.5　預測卷積層之推論時間

Predict Inference Time				
	Max	Standard Deviation	Average	
Convolutional Layer	21.9%	4.9%	9.3%	Node 1-1
	20.6%	4.3%	9.4%	Node 1-2
	21.8%	6.6%	8%	Node 1-3
	20.1%	5%	9.3%	Node 1-4

表 4.6　預測池化層之推論時間

Predict Inference Time				
	Max	Standard Deviation	Average	
Pooling Layer	12.2%	4.2%	7.2%	Node 1-1
	20%	5%	6.6%	Node 1-2
	21%	5.7%	7.3%	Node 1-3
	14.4%	4.1%	5.4%	Node 1-4

doi:10.6342/NTU201903950

表 4.7　預測全連接層之訓練時間

Predict Training Time				
	Max	Standard Deviation	Average	
Fully Connected	20.7%	5.4%	9%	Node 1-1
	26%	4%	8.7%	Node 1-2
	17.3%	5.1%	7.1%	Node 1-3
	23.5%	6.3%	8.5%	Node 1-4

表 4.8　預測卷基層之訓練時間

Predict Training Time				
	Max	Standard Deviation	Average	
Convolutional Layer	23.4%	5.9%	8.9%	Node 1-1
	16.7%	6.3%	7.2%	Node 1-2
	23.4%	4.7%	7.3%	Node 1-3
	17.9%	6.4%	9.2%	Node 1-4

表 4.9　預測池化層之訓練時間

Predict Training Time				
	Max	Standard Deviation	Average	
Pooling Layer	19%	6.2%	9.2%	Node 1-1
	20.9%	5.3%	8%	Node 1-2
	21.7%	6.6%	9.6%	Node 1-3
	18.7%	5%	8.7%	Node 1-4

而值得注意的是不論是推論或是訓練中，每一種神經網路層之最大誤差皆落

doi:10.6342/NTU201903950

在 20%左右，而這樣的誤差又是發生於什麼樣的狀況之下？

4.1.3 較高誤差項之觀察

為了解較大的誤差是如何發生的，我們做了以下的取樣如表 4.10 所表示，可以看到較大的誤差皆出先在實際執行時間較極小的例子中，例如：實際執行為 0.5 毫秒，預測時間為 0.6 毫秒，僅僅 1 毫秒的差距，但放到相對誤差中便達到了 20%以上的誤差。

表 4.10 不同參數之預測與實際之相對誤差

Input	Prediction (ms)	Actual (ms)	Error
(64, 96, 11, 32, 8, 1)	3.61	3.43	5%
(16, 512, 11, 16, 8, 2)	2	19.4	3.4%
(112, 16, 7, 2, 32, 2)	0.6	0.7	14.5%
(224, 8, 9, 32, 64, 2)	23.41	21.43	9.2%
(64, 3, 11, 4, 8, 2)	0.5	0.64	21.9%
(128, 8, 3, 1, 8, 4)	0.61	0.5	21.9%
(32, 3, 5, 64, 32, 1)	1.67	1.67	0.4%
(128, 3, 3, 2, 96, 2)	0.94	0.96	2.1%

了解到會在什麼樣的狀況下會產生較大的誤差之後，我們便做了另一項統計，以了解我們的深度學習模型對於這樣的參數所進行的預測是低估還是高估，統計的結果如下圖 4.2 與圖 4.3 表示。

doi:10.6342/NTU201903950

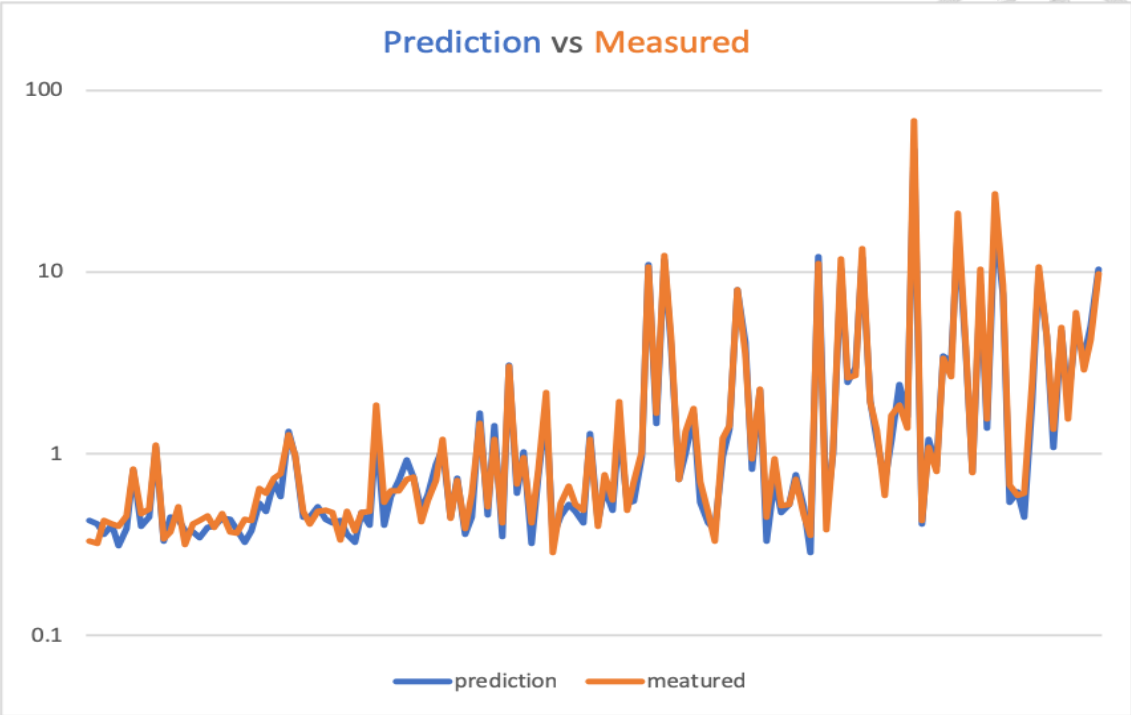


圖 4.2　不同參數之預測與實際曲線

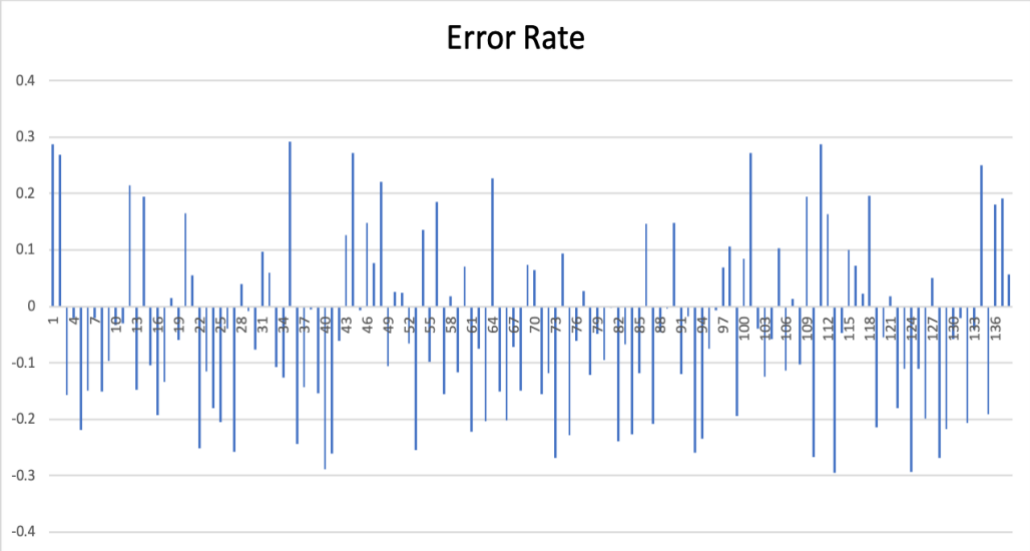


圖 4.3　圖 4.2 之各點相對誤差

在圖 4.2 中每一點皆為一組輸入參數之排列組合，縱軸為時間之對數，藍色線表示預測值；而橘色現為實際值，可以看到大部分的藍色線接在橘色線之下方，而為了更清楚地看出差別，我們整理了圖 4.3，其中橫軸為每一組的編號，縱軸則為個組之誤差，可以由圖中看出大部分的誤差皆為負值，而實際的結果也是如此，圖

doi:10.6342/NTU201903950

中包含有 100 多組參數，有 90 組的誤差皆為負值，比例大約為 70%，這表示的是我們的深度學習模型在預測時較常發生低估執行時間的狀況，那又是什麼樣的原因造成的呢？

4.1.5 TensorFlow Timeline

由於發現了上述低估時間的問題後，我們便將注意力放在硬體方面是如何去執行深度學習模型的運算，我們所使用的效能分析器為 TensorFlow Timeline，這也是 TensorFlow 所提供之 API 可以幫助我們知道在硬體中是如何實現深度學習模型的運算，包含該各個指令從何時開始計算，計算了多長時間等等的資訊皆可由此效能分析工具得到。下圖 4.4 與圖 4.5 為輸入不同參數之分析結果，兩者所測試的平台皆為 Nvidia GeForce GTX 1080，所執行的指令皆為 tf.layers.conv2d。

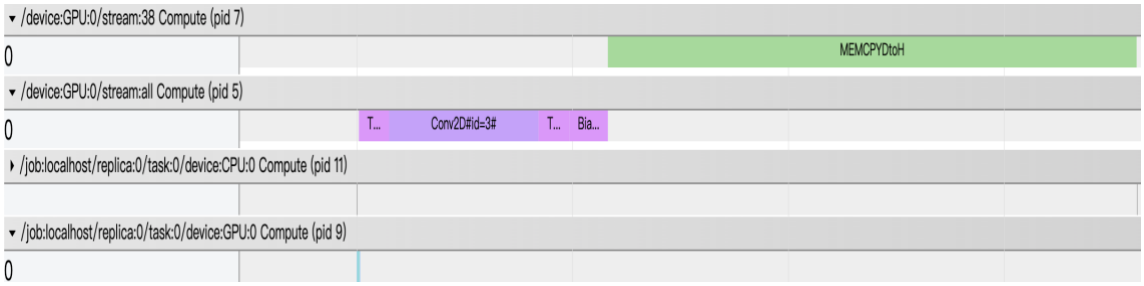


圖 4.4 TensorFlow Timeline 分析結果 1

上圖 4.4 所輸入的參數為：影像大小＝224*224、維度＝3、批次大小＝64、核心大小＝3*3、神經元個數＝96、步長＝1；執行時間為：136.15 毫秒。

doi:10.6342/NTU201903950

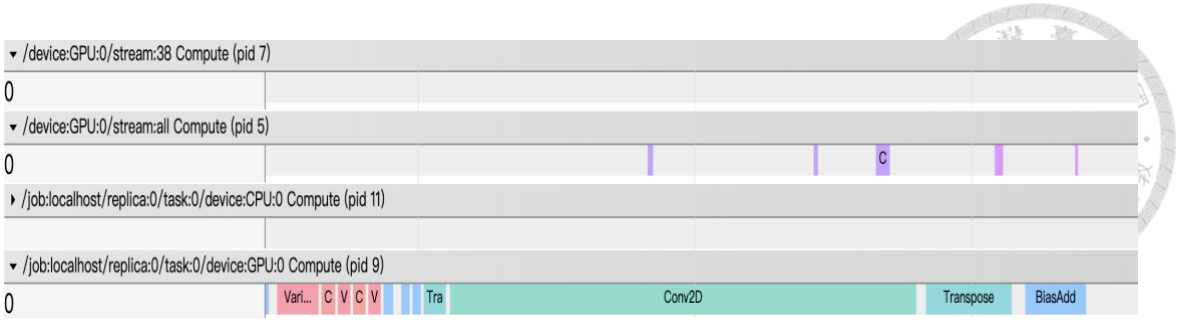


圖 4.5　TensorFlow Timeline 分析結果 2

上圖 4.5 所輸入的參數為：影像大小＝32*32、維度＝1、批次大小＝1、核心大小＝5*5、神經元個數＝6、步長＝4；執行時間為：0.55 毫秒。

而圖中的/device:GPU:0/stream 項表示為 GPU 正在執行該指令之運算；而/job:localhost/replica:0/task:0/device:GPU:0 項則表示指令執行的要求須排隊傳入 GPU 的核心之中已執行相對應之指令，由圖 4.4 可看出此項幾乎不會耗費時間，但到了圖 4.5 的狀況則完全不同。

在圖 4.4 中可以觀察出，所有的指令皆為一個接著一個在執行，但在圖 4.5 的指令與指令之間 GPU 皆存在相當大的閒置時間，其最主要的原因便為計算量的不同，在圖 4.4 大部分的時間皆是運算，因此指令排隊進入核心的時間便可以因此被覆蓋掉；而圖 4.5 則完全不同，因為其計算量過小，因此造成一個指令執行完後，下一個指令的要求還沒有傳入核心中，因此 GPU 無法執行下一個指令，必須等到進入核心後方可以執行，便因此造成 GPU 有許多的閒置時間，也可以看出圖 4.5 之執行時間基本上就等於了指令的排隊時間，並且這些排隊時間的時長範圍可以由 0.4 毫秒到 0.9 毫秒。

由 TensorFlow Timeline 的分析結果可知，在執行時間極小的軟體參數排列組合中，GPU 在進行運算時會有許多不確定時長的閒置時間，也是因為這樣的不確定，造成模型無法有效地對這樣的組合作準確的預測。

doi:10.6342/NTU201903950

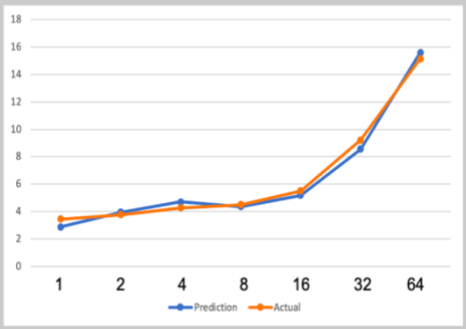
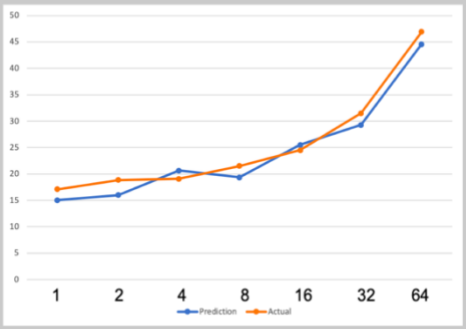
但是，雖然這樣的組合的相對誤差較大，但在絕對誤差中也都不會超過 1 毫秒，這樣的誤差其實在做整個深度學習模型的效能預測中，是非常有可能被其他種組合覆蓋掉的。

4.2 完整模型之預測

本章節將針對完整的卷積神經網路做預測，我們選用的比較基準為 TensorFlow 所提供[13]，選用的卷積神經網路包含了有：AlexNet、VGG-16、ResNet-50[14]以及 InceptionV3[15]，預測的項目包含了訓練及推論，實驗的平台為 Nvidia GeForce GTX 1080。

預測的結果如表 4.11 至表 4.14 所示，各表中之橫軸表示批次的大小；縱軸表示的為執行時間，藍色曲線為預測值；橘色曲線為實際值。

表 4.11 AlexNet

Predict Inference Time	Predict Training Time
	
Average Error: 8.9%	Average Error: 8.7%

doi:10.6342/NTU201903950

表 4.12 VGG-16

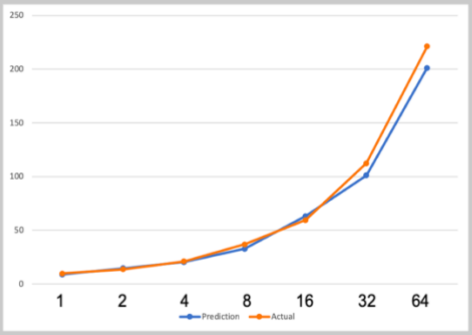
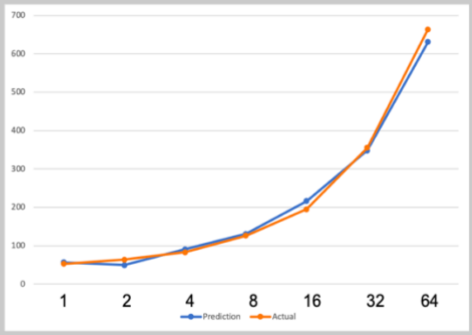
Predict Inference Time	Predict Training Time
	
Average Error: 7.8%	Average Error: 6.7%

表 4.13 ResNet-50

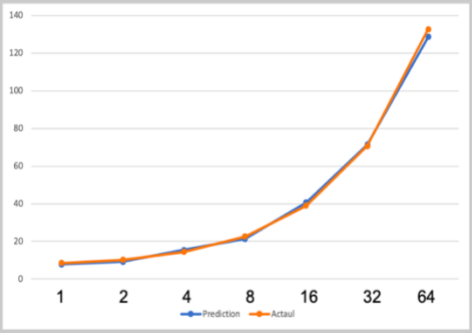
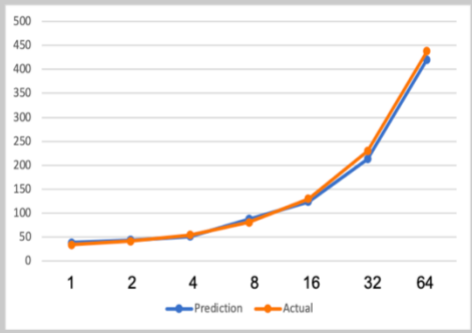
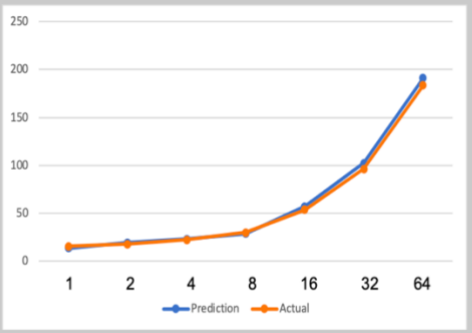
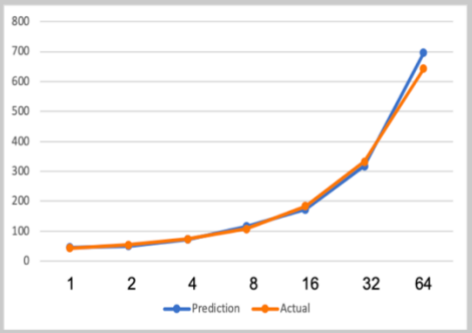
Predict Inference Time	Predict Training Time
	
Average Error: 5.4%	Average Error: 6.9%

表 4.14 InceptionV3

Predict Inference Time	Predict Training Time
	
Average Error: 6.1%	Average Error: 5.7%

doi:10.6342/NTU201903950

由上四表可看出在整個神經網路的預測中，並不會遇到前一章節所發生的問題，其原因便是在整個神經網路的運算中，執行時間極小之排列組合並不佔大多數，誤差便會因此而被均攤掉，而在表中也可觀察到，隨著批次的大小的增加，準確率也跟著上升，不僅如此，在神經網路的架構越來越大的同時，準確率也有跟著上升的趨勢，這些種種的結果皆表示了，在一整個神經網路中，即便有幾層的執行時間很小，導致預測該層執行時間時的相對誤差較大，但其執行時間主要還是有那些計算量較大的網路層所決定。

4.3 其它硬體架構之預測

本章節要做的實驗為測試我們的研究方法是否能去區分不同種類的硬體架構，下表 4.15 為本次的實驗對象，一樣包含 Pascal、Volta 以及 Turing 三種不同的架構，並且在本次的實驗中，我們增加了一個編號來區分不同的硬體架構之外，還新增了 CUDA 核心數、峰值效能、記憶體帶寬以及時脈頻率當作硬體配置的參數一起加入神經網路的訓練之中。

表 4.15 實驗配置 2

	Hardware	OS	Software
Node 2-1	Intel(R) Xeon(R) CPU E5-2620 v4 @ 2.10GHz Nvidia GeForce GTX 1080	Ubuntu 18.04	Tensorflow-GPU 1.13
Node 2-2	Intel(R) Xeon(R) CPU E5-2620 v4 @ 2.10GHz Nvidia GeForce RTX 2080 Ti		
Node 2-3	Intel(R) Xeon(R) CPU E5-2623 v4 @ 2.60GHz Nvidia Titan V	Ubuntu 16.04	

單層的神經網路預測我們選擇以預測卷積神經網路的推論時間做呈現，其中我們訓練的平台為 Nvidia GeForce GTX 1080，訓練的結果如下表 4.16 所示，由表

doi:10.6342/NTU201903950

中的結果可以看出，只要在訓練時輸入的參數足夠，我們的神經網路是可以分辨出不同的架構並且使用各架構所對應之公式去做計算。

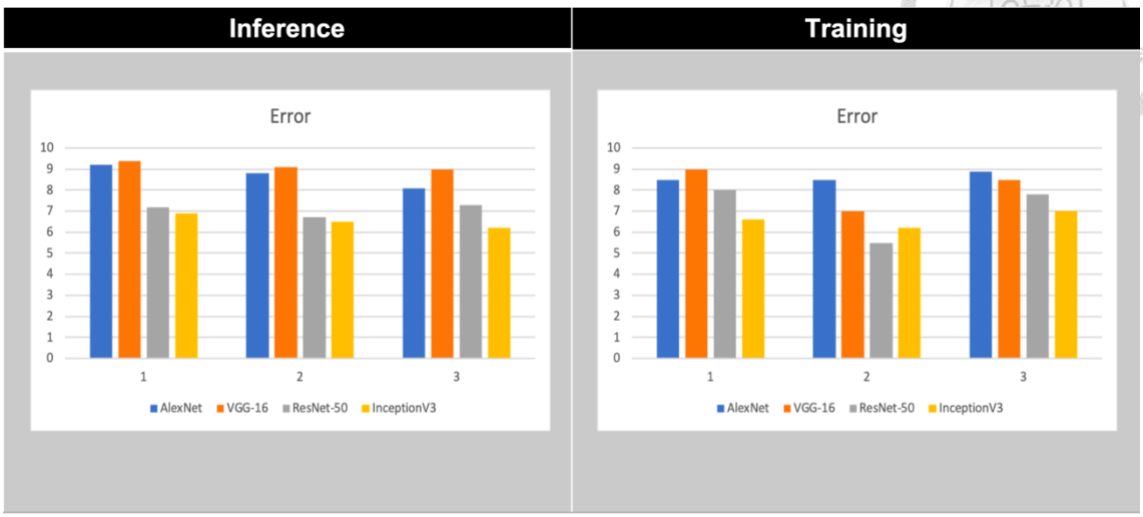
表 4.16 預測不同硬體架構之卷積層推論時間

Predict Inference Time				
	Max	Standard Deviation	Average	
Convolutional Layer	22.7%	4.1%	6.2%	Node 2-1
	19.1%	6.8%	8.4%	Node 2-2
	19.2%	5.7%	7.6%	Node 2-3

而預測完整的神經網路則如下表 4.17 所示，表中的橫軸表示對不同的硬體做的預測，縱軸為相對誤差。由此表中的縱軸最高值可觀察到，對於所有卷積神經網路之執行時間的預測準確率接在九成以上，並且由表也可以較清楚的看出隨著神經網路架構的成長，誤差皆有下降的趨勢，其中的原因便與前一章節所提到的相同，即隨著神經網路的成長，其中預測誤差較大的那幾層在做運算時所佔的比例並不高，因此即便對其預測的誤差較大，但只要能夠準確的預測該神經網路中計算量較大的幾層，即可準確預測整個網路的執行時間。

doi:10.6342/NTU201903950

表 4.17 預測不同硬體架構之神經網路執行時間



4.4 未取得硬體之預測

此實驗為本研究之最後一項實驗，同時也是最重要且最最有挑戰性之實驗，本實驗的目標為預測未取得之硬體效能，我們選用的機器如下表 4.18，其中包含了五種 GPU，所有的 GPU 皆同為 Pascal 架構之加速器，其中各機器主要的差異在於 CUDA 核心數、記憶體帶寬、峰值效能以及時脈頻率種種硬體參數配置，因此在執行上會有效能上的差異。而這些參數也一併被加入到我們所建之深度學習模型中當作學習的資料。

doi:10.6342/NTU201903950

表 4.18 實驗配置 3

	Hardware	OS	Software
Node 3-1	Intel(R) Xeon(R) CPU E5-2620 v4 @ 2.10GHz Nvidia GeForce GTX 1080	Ubuntu 18.04	Tensorflow-GPU 1.13
Node 3-2	Intel(R) Xeon(R) CPU E5-2620 v4 @ 2.10GHz Nvidia GeForce GTX 1070		
Node 3-3	Intel(R) Core(TM) i7-6700TE CPU @ 2.40GHz Nvidia Quadro P1000		
Node 3-4	Intel(R) Core(TM) i3-6100TE CPU @ 2.70GHz Nvidia Quadro P2000		
Node 3-5	Intel(R) Core(TM) i3-6100TE CPU @ 2.70GHz Nvidia Quadro P4000		

這裡我們選擇呈現預測單層卷積神經網路之訓練時間的結果，如下表 4.19，其中最後一行的硬體表示為：訓練模型時所使用的資料不包含此機器之資料 。以最後一行之 Node3-1 為例，其代表的意思為，在訓練的過程中，其中訓練的資料只包含了 Node 3-2 至 Node 3-5 之參數以及執行時間，而到了推論時，也僅僅只有輸入 Node 3-1 之參數。以此類推，其他列亦是如此。

而由表 4.19 之結果可知，預測結果均不如先前所呈現，主要還是因為其中雖然各硬體皆為相同的架構，但其效能上的差異並不是呈線性關係，因此在訓練資料有限的狀況下其訓練的準確度也有限。雖然說準確度無法達到前幾章節所顯示，但其準確率依舊有達到八成以上的效果。而接下來將呈現對於一整個神經網路的預測結果。

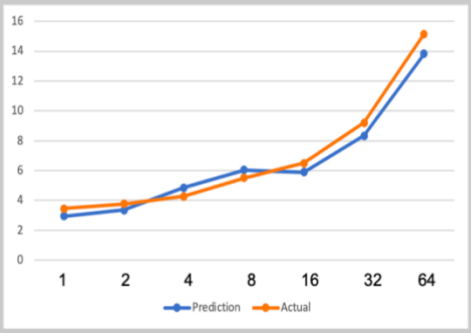
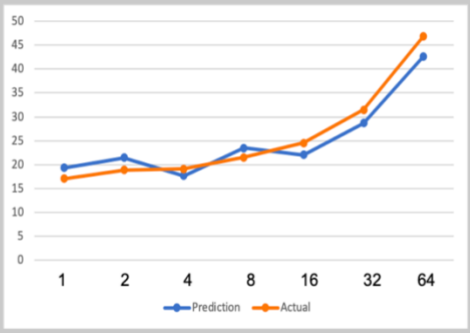
doi:10.6342/NTU201903950

表 4.19 預測未取得之硬體之卷積層訓練時間

Predict Training Time				
	Max	Standard Deviation	Average	
Convolutional Layer	25.6%	9.8%	15.4%	Node 3-1
	26%	10.5%	14.6%	Node 3-2
	25.3%	6.3%	13.8%	Node 3-3
	24.5%	7%	12.3%	Node 3-4
	22.6%	9.1%	14.2%	Node 3-5

下表 4.20 至表 4.23 為對於未取得硬體之完整卷積神經網路之預測，我們所預測的平台為 Nvidia GeForce GTX 1080，可以觀察到其準確率依舊不如前段章節所呈現，但其總體而言也有九成左右的準確率，而由於我們在收集資料的同時，皆有對計算量較大之神經網路層做特別的優化，因此觀察到隨著神經網路的更深及輸入的圖片更大的同時，其中預測的準確率也跟著上升。

表 4.20 AlexNet (未取得之硬體)

Predict Inference Time	Predict Training Time
	
Average Error: 11.1%	Average Error: 10.2%

doi:10.6342/NTU201903950

表 4.21 VGG-16 (未取得之硬體)

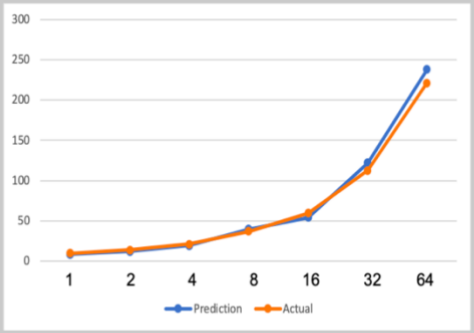
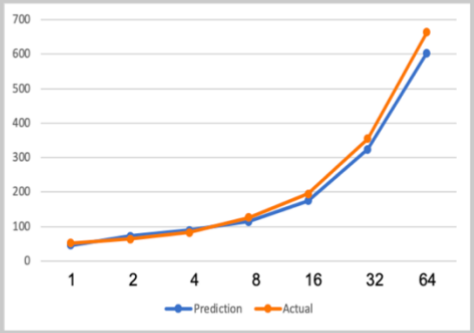
Predict Inference Time	Predict Training Time
	
Average Error: 9.8%	Average Error: 10.2%

表 4.22 ResNet-50 (未取得之硬體)

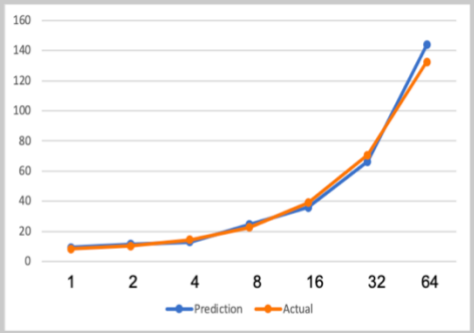
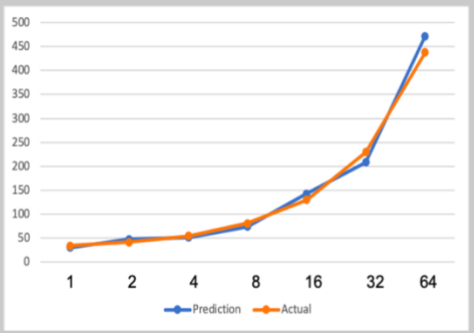
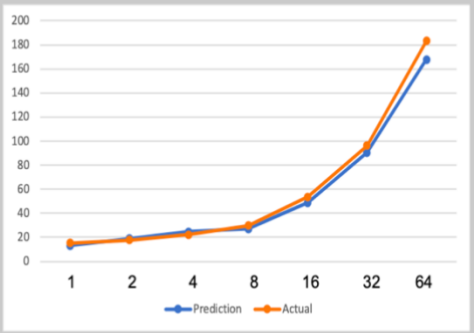
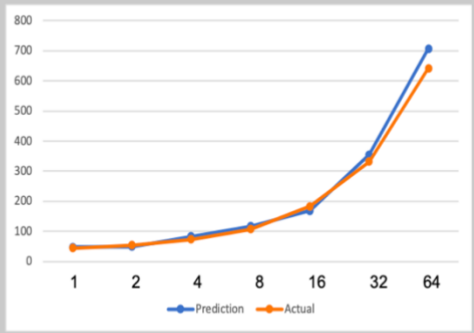
Predict Inference Time	Predict Training Time
	
Average Error: 9%	Average Error: 9.8%

表 4.23 InceptionV3 (未取得之硬體)

Predict Inference Time	Predict Training Time
	
Average Error: 9%	Average Error: 9.3%

doi:10.6342/NTU201903950

由於在實作 TensorFlow 所提供之 Benchmark 時，其中數據的呈現方式皆是以吞吐量也就是執行效率來表示，因此下表 4.24 與表 4.25 便是以相同的方式去做呈現。

表 4.24 推論之吞吐量預測

Batch Size=32			
	Prediction (image/sec)	Actual (image/sec)	Error
AlexNet	**3156.01**	**3475.78**	**9.2%**
Vgg-16	**308.85**	**284.39**	**8.6%**
ResNet-50	**422**	**453.28**	**6.9%**
InceptionV3	**356.11**	**332.5**	**7.1%**

表 4.25 訓練之吞吐量預測

Batch Size=32			
	Prediction (image/sec)	Actual (image/sec)	Error
AlexNet	**932.14**	**1016.51**	**8.3%**
Vgg-16	**100.19**	**92.09**	**8.8%**
ResNet-50	**132.38**	**139.49**	**5.1%**
InceptionV3	**89.24**	**96.27**	**7.3%**

doi:10.6342/NTU201903950

第五章　　結論與未來目標

由本研究可以知道，在做效能預測時，即便是使用公式推導的方式來做預測，其中執行時間並不會單純地與批次大小呈現線性關係，這點即是使用公式法最明顯也是最致命的缺點。

而我們所提出的：以深度學習的方式來做預測，雖然說我們的方式需要有收集資料的步驟，但可以看出在單一機器的情況下皆可準確的預測出結果，不論是推論或是訓練、單層亦或是多層。

且這樣的方法不僅僅只針對單一機器，在實驗中我們也將其拿來預測不同機器甚至是做不同架構之間的預測，並且也得到了準確的結果，因此這也表示我們的模型是有足夠的能力去區分不同的架構並針對不同的硬體做不同的預測。

最重要的是，我們的模型亦可以對新的或是未取得的機器之效能做簡單的概算以及分析。藉由深度學習的方式，我們雖然花了部分時間收集足夠的資料及訓練模型，但其為我們省去的是研究底層硬體的計算，並且在日後的有新的機器或是模型被提出時，我們要做的也只是調整輸入模型的參數，便可在第一時間對其做簡易的效能分析。如此一來，便可以幫助日後的使用者可以快速地挑選所需要的加速裝置或者是深度學習模型的參數配置。

雖然本研究可以對未取得之機器之效能做簡單的預測及分析，但我們依舊希望在日後可以將其準確率更加的提升，最理想的狀況即是將其提升至和單一機器之預測結果相同。

另外隨著深度學習的發展，單一加速器的裝置效能越來越無法滿足日新月異的應用，因此許多人開始將其應用投入到分散式的裝置上執行以提高效率，因此本

doi:10.6342/NTU201903950

研究也希望可以對分散式裝置做相關的預測及分析，不論是單一機器上的多台加速器亦或是以網路連接之多台加速器。

另外，現在也有越來越多的異質加速平台，而我們的目標也不僅僅是放在個人電腦的效能預測上，我們希望能將這樣的預測方式移植到嵌入式系統上，像是 TX2、FPGA…等等不同的機器，甚至是 TPU 也是我們日後的研究目標，並且也希望能夠將不同的深度學習之軟體架構像是 PyTorch 或是 Keras。

甚至是加入更多樣的硬體參數像是 GPU 的使用率，此參數的使用場境主要應用於神經網路有分支時。例如：神經網路有 A 與 B 兩分支，其中單獨執行 A 會花上 1 秒且使用率為 60%；單獨執行 B 會花上 0.8 秒且使用率為 50%。以上述的例子，若是今天神經網路並不是以 Critical Path 執行而是同時執行兩分支時，是否可以利用執行時間與 GPU 使用率這樣的參數搭配推導出一個公式，並以此在日後可以更好的預測更加複雜的神經網路。

隨著深度學習的快速發展，這樣的快速效能分析工具的研究也會愈發重要，因此我們所提出的方式不僅所需要的步驟及背景知識不多外，這樣的方式同時也是可模組化的，對於使用者而言可以輕鬆地增加或減少訓練時所需的參數已訓練出自己的效能預測之深度學習模型。而我們也希望本研究可以幫助許多公司在做深度學習的應用時，可以使用我們的工具快速篩選所需要購買的加速裝置甚至是決定其他的硬體配置。

doi:10.6342/NTU201903950

參考資料

[1] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” Nature, vol. 521, no. 7553, pp. 436–444, 2015.

[2] H. Qi, E. R. Sparks, and A. Talwalkar, “Paleo: A performance model for deep neural networks,” 2016. [Online]. Available: https://openreview.net/pdf?id=SyVVJ85lg

[3] https://github.com/TalwalkarLab/paleo

[4] C. Coleman, D. Narayanan, D. Kang, T. Zhao, J. Zhang, L. Nardi, P. Bailis, K. Olukotun, C. Re´, and M. Zaharia, “Dawnbench: An end-to-end deep learning benchmark and competition,” *Training*, vol. 100, no. 101, p. 102, 2017.

[5] https://mlperf.org

[6] https://en.wikipedia.org/wiki/Convolutional_neural_network

[7] https://en.wikipedia.org/wiki/Long_short-term_memory

[8] “Tensorflow,” https://www.tensorflow.org/

[9] Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXivpreprint arXiv:1409.1556.

[10] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems (pp. 1097-1105).

[11] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. (1998).Gradient-based learning applied to document recognition. In Proceedings of the IEEE (pp. 2278–2324).

[12] http://www.image-net.org

[13] https://github.com/tensorflow/benchmarks

doi:10.6342/NTU201903950

[14] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2818-2826).

[15] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).

doi:10.6342/NTU201903950