**Youbike站點存量數據訓練**

**GMAN預測模型**

組員

資碩工二 109753101 賴玠忠

資碩工一 110753117 莊崴宇

資碩計一 110753202 何彥南

**目錄**

一、緒論(含研究背景) P.1

二、研究動機與目的 P.2

三、研究方法 P.3-12

第一節、研究架構 P.3

第二節、資料收集 P.3

第三節、資料前處理 P.3-4

第四節、模型介紹 P.5-7

第五節、實驗設計 P.7-8

第六節、實驗結果 P.8-12

四、研究結果 P.13

五、結論與未來展望 P.14

六、程式碼 P.15

七、參考文獻 P.15

1. **緒論**

結合公車、捷運等轉乘優惠，Youbike近年逐漸成為現代人經常用於短程代步的交通工具，捷運站出口、學校附近等，設立越來越多租借站供應民眾使用，不僅提高附近景點的交通便利性，也可以減少部分顛峰時間的車流問題。

然而，隨著使用率的提升，相關問題也隨之浮現，除了輪胎、座椅、零件損毀等腳踏車問題，關於管理層面的問題也逐漸產生。在生活中我們經常會遇到想要租車卻沒有車可以借的情形，透過增加補給次數即可以解決此問題，然而；停車時卻經常也遇到沒有空位可以還車的問題產生，上述兩個問題都是由於Youbike補給系統為規劃完善導致，因此本實驗決定以深度學習模型預測Youbike租借站於下一時間點的腳踏車站點存量，針對預測的結果即可以對該時間點的補給情形進行調整，若預測結果為0的站則應該多補給腳踏車，預防民眾沒有車借的情形，而若預測結果過多的站則應該至該站撤回幾台腳踏車，預防民眾沒有空位還車的情形產生。

以Youbike數據結合深度學習做站點存量結果預測的文獻在過去已出現多次，大多預測僅考量空間或是時間因素進行預測，結果並未如預期中理想，至今，站點存量規劃的問題依舊存在，我們認為，若要進行交通類型的預測，必須同時考量空間和時間因素，因此我們透過網路上相似的知名數據集研究，探討關於相關數據集的預測研究文獻，套用文獻中的模型於本次Youbike站點存量資料，希望能有更近一步的改善。

最終，我們透過我們的目的與需求，選擇Benchmark中排名第三名的GMAN時間序列模型作為我們訓練的預測模型。

1. **研究動機與目的**

由於近期Youbike使用者人數逐漸增加，Youbike租借站的數量也隨之而增，過往的站點大多集中於學校、捷運等，人口較多且離公車站牌也較相近的地點，方便民眾作為於短程交通的代步工具，如今，不僅是上述等人口較多的地點，甚至是住家附近、公園等，也都逐漸設立的Youbike站點供應民眾租借，方便民眾搭乘完大眾運輸交通工具騎乘Youbike回家，因此租借的需求大幅提升。然而，隨著租借頻率增加，在補給系統出現了規劃方面的問題，許多使用者在想要租車時卻會發現該租借站沒有車可以騎乘，可能是剩下幾台故障的車，也可能完全沒有剩餘的車可以使用，同時，在租借頻率較少的時段，經常發現租借站停滿了車，完全沒有空位可以讓該時段的使用者歸還，只能選擇延長租借時間騎往附近的其他租借站還車。本應該是為了交通方便而設立的Youbike卻因為補給問題造成服務目標本末倒置，反而讓使用者花費更多時間在處理等車或等空位的問題，透過這次機會，我們想將機器學習應用於該方面，從中談討上述問題的原因以及嘗試找出其解決方法。

本研究旨在同時考量時間和空間因素，選擇以時間序列性模型針對不同時間點的Youbike站點存量進行預測，以預測下一時間點不同站點的腳踏車數量，利用預測結果規劃Youbike的補給系統問題，以不同站點於不同時間點的Youbike租借情形對補給系統調整，解決民眾使用Youbike時所遇到數量不足而沒車租借或是還車時空位不足而無法歸還的問題，藉此提升使用Youbike的便利性。

1. **研究方法**

**第一節、研究架構**

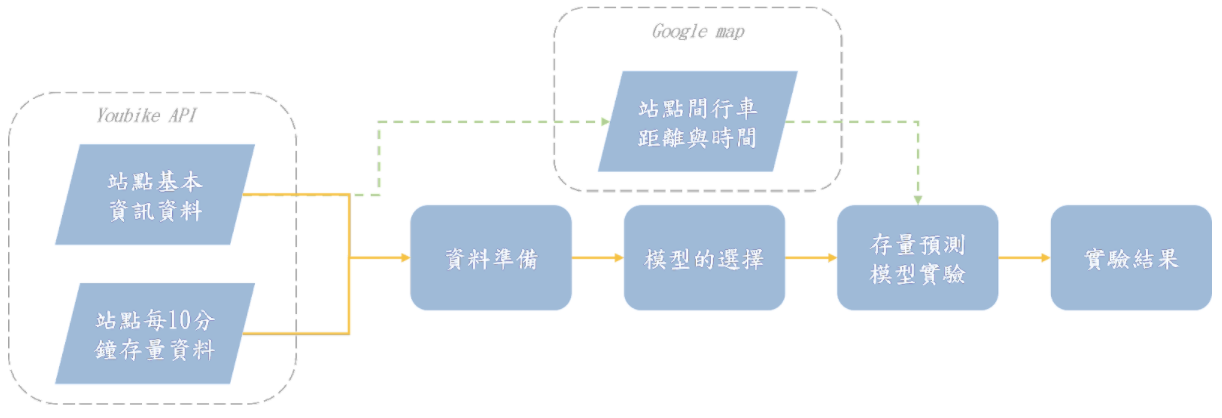
整體研究流程會先以資料講解開始，敘述數據的由來和前處理流程，再講解GMAN模型的選用原因和模型整體的運算原理、流程，接著針對實驗規劃和實驗結果的評估標準進行解說，最後從實驗結果中探討結論、改善方法以及未來展望，如下方流程圖。

圖1. 研究流程圖

**第二節、資料收集**

實驗數據來源藉由台北市公開的Youbike即時存量api，蒐集自2021年12月4日至2022年1月8日，共36天內，各Youbike站點每10分鐘的腳踏車存量資料，其中包含台北市中的12個行政區，共399個站點。

**第三節、資料前處理**

爬取資料後進行初步清整，將原先下載資料由json檔格式的資料轉換成兩種資料，分別為站點基本資訊的資料共399筆，每10分鐘站點存量變化的資料共76210筆資料。表1與表2為資料欄位資訊，圖2為爬取到台北市各行政區的站點位置。本次專案因GMAN模型多站點預測多站點架構影響，過多的站點會導致GPU空間不足，因此本實驗先以文山區的資料進行實作，扣除期間沒營運的站點，其中包含26個站點。

一張含有 地圖 的圖片

自動產生的描述圖2. 台北市Youbike站點示意圖

表1. 欄位資訊表 表2.欄位資訊

(每10分鐘站點存量變化資料) (站點基本資訊)

|  |  |  |
| --- | --- | --- |
| 欄位名稱 | 中文名稱 | 是否使用 |
| sno | 站點編號 | ✔ |
| sna | 站點名稱 | ✔ |
| tot | 車位總量 | ✔ |
| sarea | 所屬區域 | ✔ |
| lat | 緯度 | ✔ |
| lng | 經度 | ✔ |
| ar | 站點地址 | ✘ |
| sareaen | 所屬區域(英文) | ✘ |
| snaen | 站點名稱(英文) | ✘ |
| aren | 站點地址(英文) | ✘ |

|  |  |
| --- | --- |
| 欄位名稱 | 中文名稱 |
| sno | 站點編號 |
| sbi | 目前車輛數目 |
| bemp | 目前空位數量 |
| act | 站點是否啟用 |
| date | 爬取日期 |
| time | 爬取時間(10分鐘/筆) |

**第四節、 模型介紹**

1. **模型選擇**

依照本實驗的資料類型，我們以時間序列去預測對數據進行分類，多個站點間彼此有關聯性(圖3)，因此我們找到兩個在交通流量的時間預測上著名的外國資料集做為參考:

1. METR-LA.－該交通數據集包含從洛杉磯縣高速公路環路檢測器收集的交通信息。選取207個傳感器，採集2012年3月1日至2012年6月30日4個月的數據進行實驗。
2. 一張含有 地圖 的圖片

   自動產生的描述PeMS-BAY.－該交通數據集由美國加州運輸局(CalTrans)績效測量系統(PeMS)收集。在灣區選取325個傳感器，收集2017年1月1日至2017年5月31日6個月的數據進行實驗。

圖3. 兩資料的車流感應器分布

一張含有 文字, 監視器, 螢幕, 電視 的圖片

自動產生的描述透過上述兩資料集在 PaperWithCode 上的 benchmark 去尋找適合、創新且有公開pytorch程式碼的深度學習模型，最後我們選擇在PeMS-BAY Benchmark上表現第三名的GMAN模型，同時也是有提供程式碼的模型中表現最好的預測模型，如圖4。

圖4. Benchmark排名示意圖

1. **模型介紹**

GMAN模型中主要有兩大特點，分別為Encoder and Decoder技術和結合Spatio-Temporal Embedding 的Multi-Attention值運算，以下分別針對兩個部分進行講解。

1. **Encoder - Decoder**

當輸入的時間與預測的時間長度不同時，Encoder - Decoder 的架構可以讓模型透過中間的 TransAtt更好的轉換(圖5)，不會因為 input 和 output 的序列長度不同導致模型預測時的偏差。而同樣的技術也用在 Seq2seq，可以用在翻譯上，因為在翻譯任務上常常會遇到輸入的字數和翻譯目標的字數不一樣。

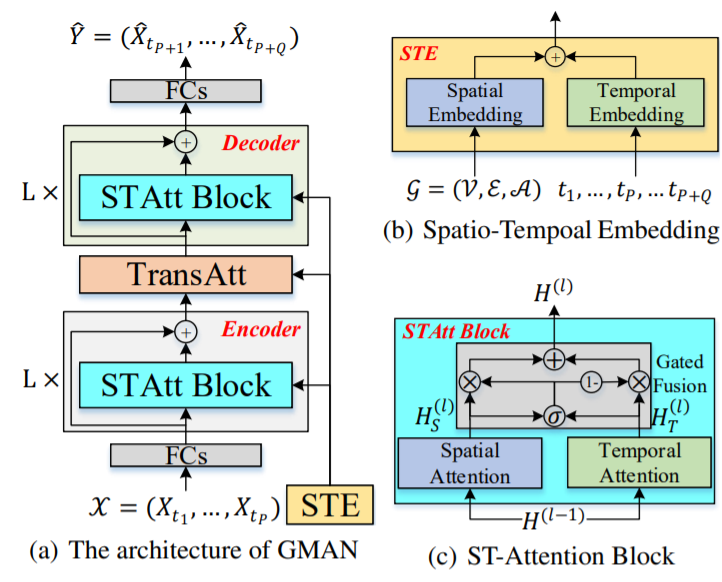


圖5. GMAN 模型架構圖

1. **Multi-Attention and Spatio-Temporal Embedding**

深度學習中 Attention 機制是一個重要的技術，最早是由AshishVaswani等人發表的Attention is all you need中提出的。間單來說它的概念就是在隱藏層上再加一層可以讓魔興決定要給予隱藏層中的哪個輸出更高的權重，透過疊加好幾層模型可以在不同層中學習到要專注在哪個特徵上藉此優化預測。而 multi-Attention 的意思就是將多個attention並行在一起處理，最後在 concat 在一起做預測。雖然會增加訓練時的參數和時間，但是對模型會更好去學習更深一層的資訊。

STE 的部分由SE和TE 組成，SE部分是使用站點間直線距離作為edge的值，並使用史丹佛提出的node2vec將圖的資訊轉換成每個點固定維度的特徵序列，方便後續進入到深度學習做訓練。而TE的部分則是將一天分成T個時間區間，並將每筆時間序列資料的時間部分轉換成在一天中的哪個時間區間，還有在一個禮拜中的星期幾，將此資訊用one-hot的方式編碼，並concat作為 TE。

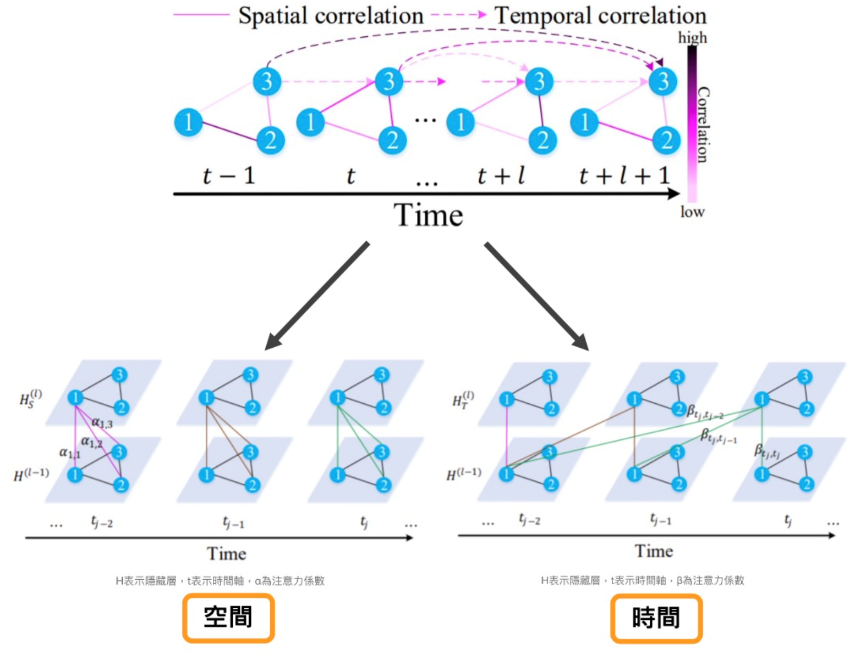
GMAN模型則是利用此特性同時對輸入的存量資料與STE去訓練，並利用attention機制讓模型訓練時能對不同的時間和不同的地理關係中找到重要的部分去訓練(如圖6)。

圖6. GMAN 模型中地理與時間上的注意力機制示意圖

**第五節、 實驗設計**

1. **CPU、GPU效率比較**

分別以CPU(Intel Core i7-10710U processor)、GPU(NVIDIA GeForce GTX1650/4GB GDDR5)、Tesla K-80 GPU訓練模型，比較不同情況下的模型訓練時間差異。其中CPU 與 GPU 是在自己筆電上測試， Tesla K-80 是在 colab 上做測試。

1. **超參數訓練比較**

調整Learning Rate和Batch Size，評估模型適合的超參數值，以作為最終模型之超參數設定，Learning Rate以0.1、0.01、0.001作為測試值，而Batch Size以1、5、10、20、30、40、50、60作為測試值。

1. **Adj門檻值設定**

考量點和點之間的空間關聯性，以兩站點間的直線距離換算為Adj值，換算後的Adj值與實際距離成反比，如圖7，排除直線距離過遠的站點組合，透過設定不同門檻值評估訓練結果，以整體資料Adj值的0、25%、50%、75%、平均值作為測試值。

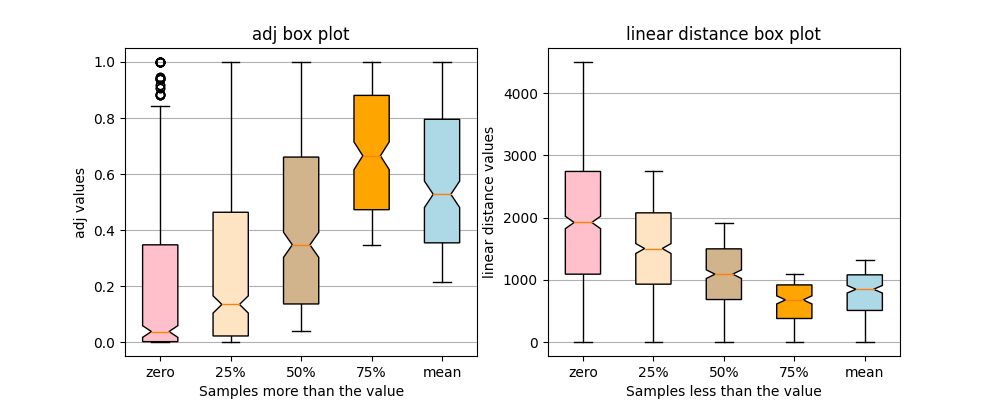


圖7. Adj值與直線距離關係圖

1. **History Step設定**

本實驗以前n個時間點的站點存量資料預測下一筆的站點存量，透過設定不同n值測試，選擇較佳的n值作為最終模型之參數設定。

1. **模型評估值介紹**

本實驗分別以GMAN文獻中所提及的MAE、RMSE和MAPE作為上述四個實驗結果的評估標準，並以MSE作為最終模型和預設模型的Loss值評估標準。

**第六節、 實驗結果**

1. **CPU、GPU效率比較**

由實驗結果可以看出，使用GPU進行模型訓練所花費的時間大約比使用CPU訓練的一半高一些，節省相當多時間，實驗結果如圖8。

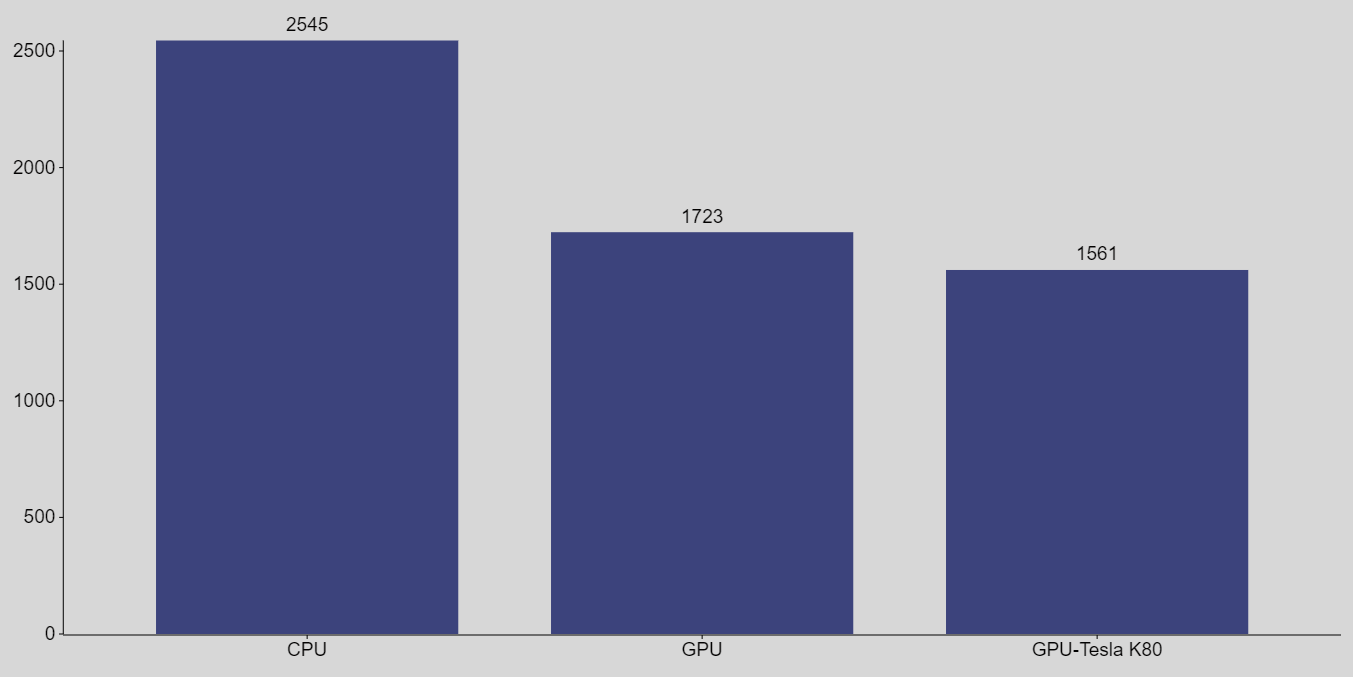


圖8. CPU、GPU效率比較圖

1. **超參數訓練比較**

**(1) Learning Rate**

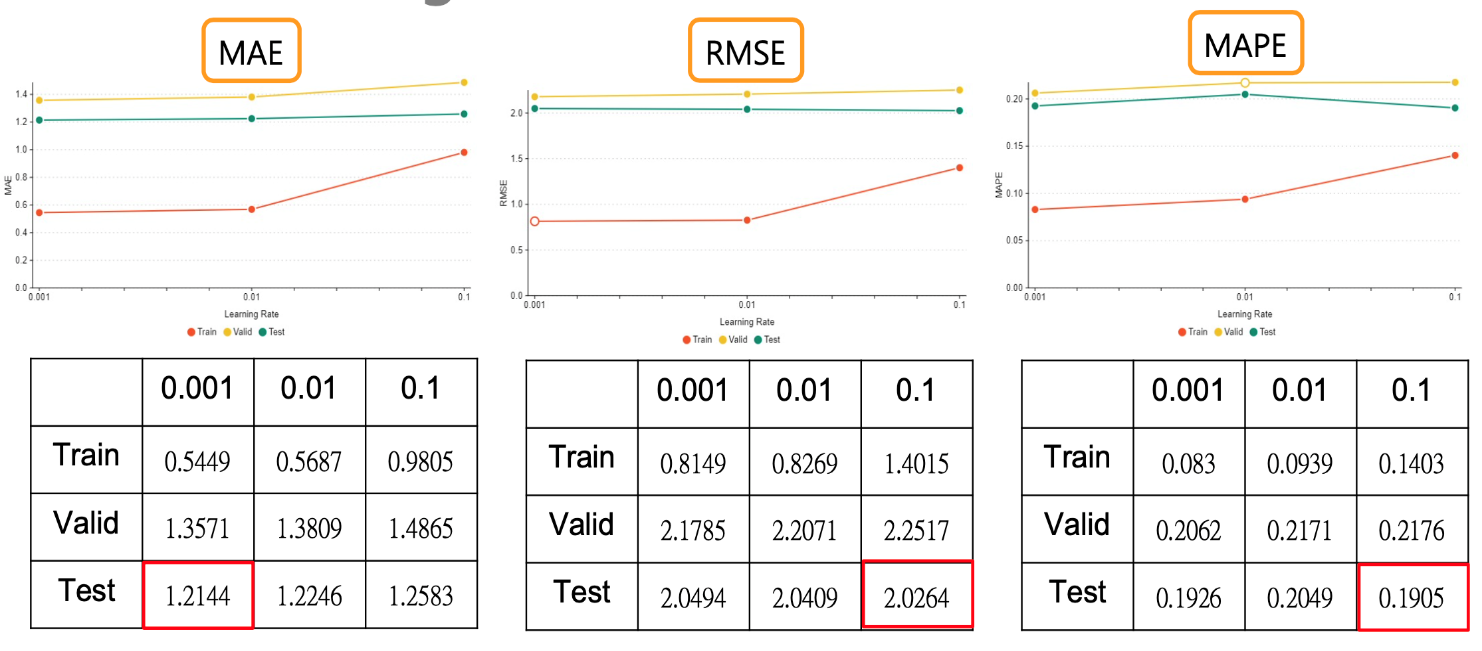
由圖9可以看出Learning Rate設定於0.001或0.1時，整體模型有最小的誤差值，然而，透過文獻參考得知最佳的Learning Rate通常介於0.01至0.0001之間，因此我們藉由MSE的Loss值曲線圖判斷應將Learning Rate設定為多少。透過圖10可以看出，當Learning Rate設定於0.001時，整體的Loss值相較設定為0.1時穩定，因此選用Learning Rate為0.001作為最終模型之超參數。

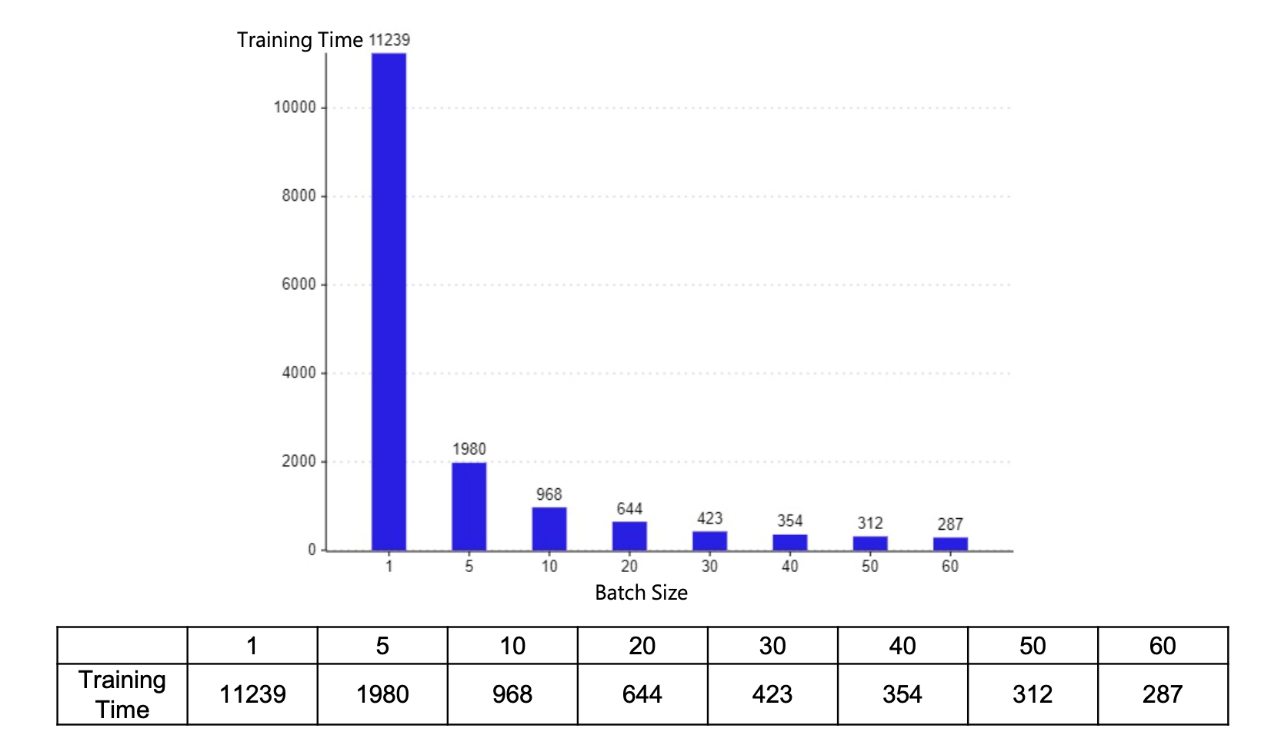
圖9. Learning Rate之誤差值比較圖(綠色為Test Set)

圖10. Learning Rate之MSE Loss值比較曲線圖

(藍色為Train Set結果、紅色為Validation Set結果)

**(2) Batch Size**

Batch Size設定為5時，該模型於MAE、RMSE、MAPE三個評估值中皆為最低誤差值，由圖11可得知。此外，由圖12可以得知，當Batch Size設定為1時，會花費過多時間於訓練模型，設定為5以上可以大幅降低訓練時間，因此我們選用以Batch Size為5作為最終模型之超參數設定。

一張含有 桌 的圖片

自動產生的描述圖11. Batch Size之誤差值比較圖(綠色為Test Set)

圖12. Batch Size之Training Time比較圖表

1. **Adj門檻值設定**

表3為整體資料之Adj值統計資料，圖13為不同Adj門檻值之誤差值測試實驗結果，由該圖可得知，當Adj門檻值設定為25%，也就是0.00124時，該模型有最小誤差值，因此選用25%作為最終模型之Adj門檻值。

|  |  |
| --- | --- |
| Mean | 0.20827 |
| std | 0.29777 |
| min | 0.00000 |
| 25% | 0.00124 |
| 50% | 0.03739 |
| 75% | 0.34703 |
| MAX | 1.00000 |

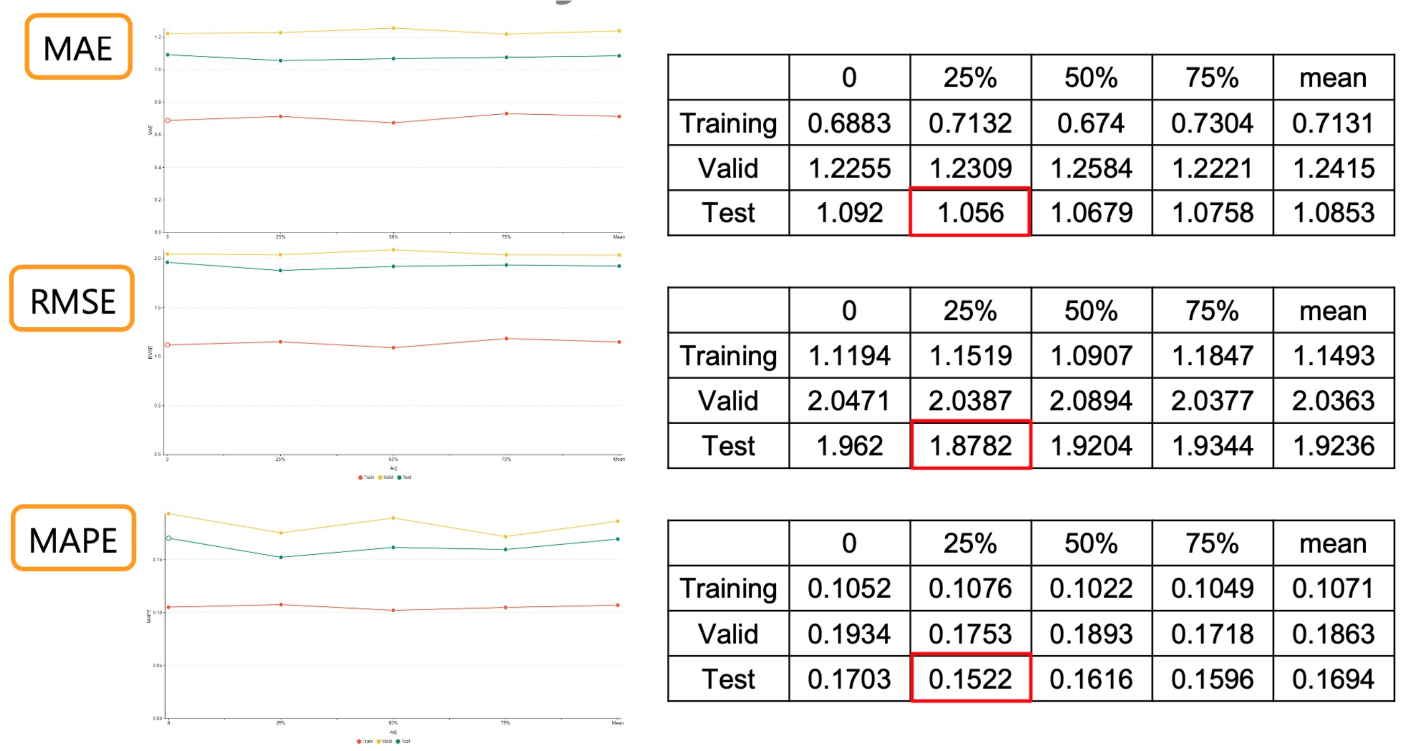
表3. Youbike站點之Adj統計值

圖13. Adj門檻值之誤差值比較(綠色為Test Set)

1. **History Step設定**

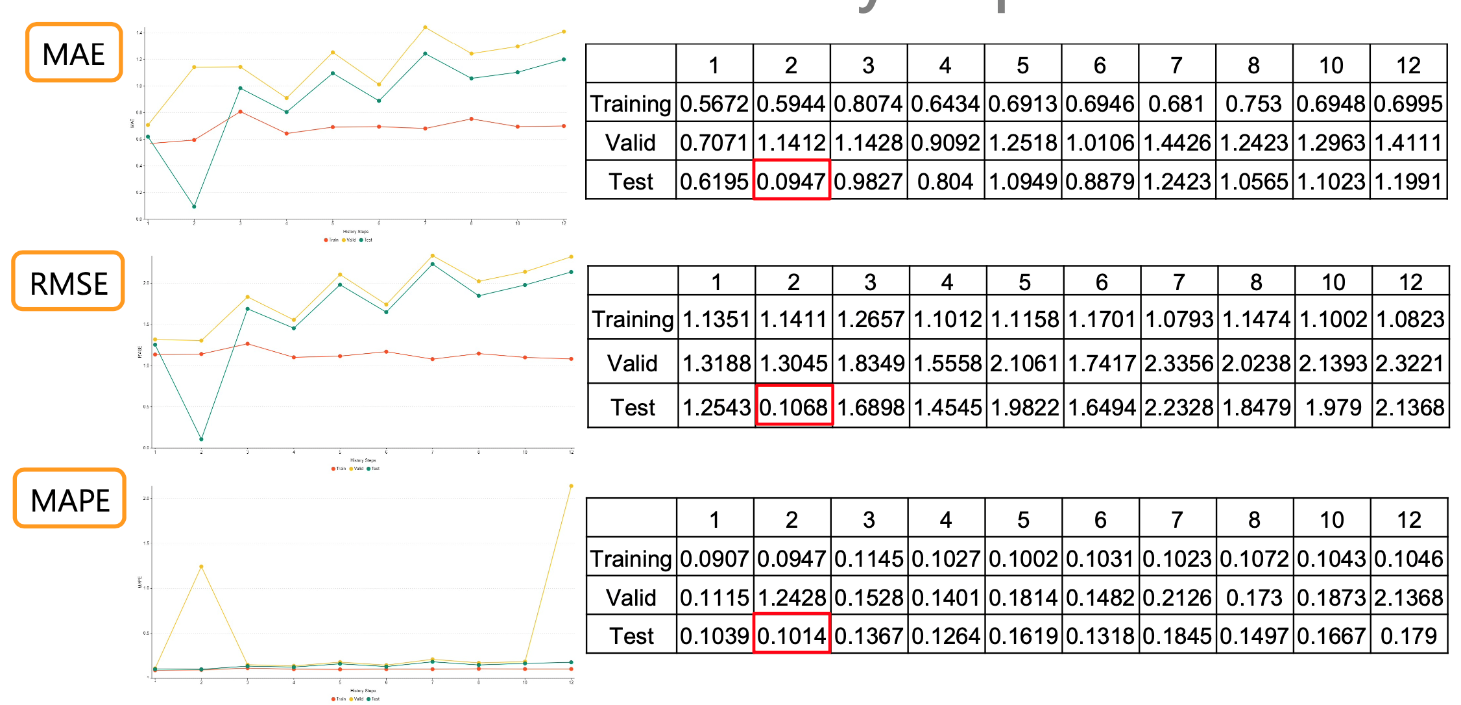
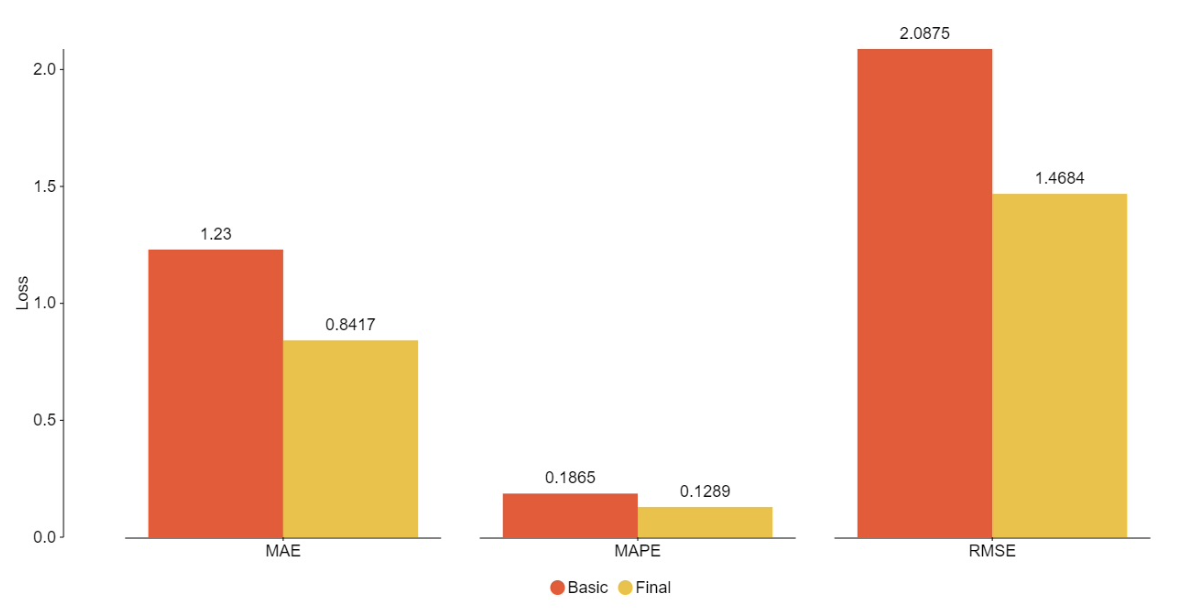
由圖14可以得知，當History Step設定為2時，該模型有最小的誤差值，因此我們將最終模型之History Step參數設定為2。

圖14. History Step之誤差值比較圖(綠色為Test Set)

1. **研究結果**

整合第三章第六節的實驗結果，我們最終的GMAN預測模型以Learning Rate = 0.001、Batch Size = 5、Adj門檻值設定=25%(0.00124)、History Step = 2設定，並將此最終模型與尚未調整過的預設模型透過MSE之Loss曲線圖比較，呈現實驗之最終成果。由圖15和圖16可以看出有調整過後的最終模型之MAE、RMSE、MAPE三種評估值以及MSE曲線圖中的Loss有明顯下降的趨勢，原本整體的Loss值應由3點多開始收斂，而經過調整，最終模型從二點多開始收斂，由此可證，模型訓練有改善該模型之預測效果。

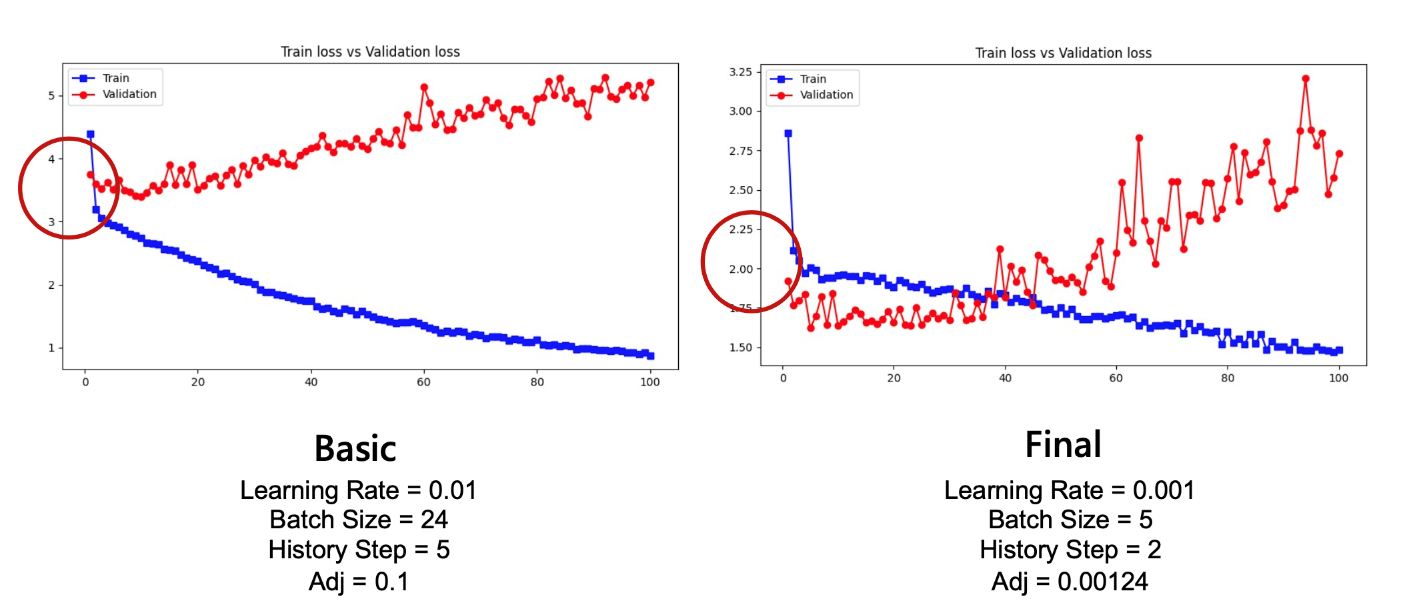
圖15. 預設模型與最終模型之MAE、MAPE、RMSE值比較圖(黃色為最終模型)

圖16. 預設模型與最終模型之MSE Loss值比較曲線圖(右邊為最終模型)

1. **結論與未來展望**
2. **結論**

在本次的期末專案中，透過爬蟲獲取Youbike 在不同存量資料，並透過 Benchmark 找尋適合與開源的程式碼去實作多點的時間序列預測。最後設計四種實驗去探討效率與調整模型。

在訓練的速度上，我們發現在訓練參數量與資料量不變的情況下，影響訓練速度的因素主要是在於GPU的使用與batch size的設置。GPU 的訓練速度真的是比 CPU快很多。而在batch size 則是越大越訓練越快，但越大代表會使用越多記憶體，所以還是有個上限。而在訓練效果上，batch size 不是越高越好，實驗中batch size 在5時最好。Learning rate 雖然效果影響不明顯，但以模型的穩定度來看在 0.001時最好。在 adj 門檻的設置上，我們發現在 Youbike 資料上，設為 25百分位數的值時效果最好，也就是說在所有26站點組成的 edge 中只要看前 75% 關係大的的 edge 就好了，剩下的可以忽略。最後我們以不同時間長度點作為輸入，同樣去預測下個時間點的存量，我們發現2的時候效果最好，由此可知對模型來說前20分鐘的歷史存量資料就足以判斷後10分鐘的存量。

綜合上述實驗，我們將最後調整的 final 模型與預設參數的 basic 模型做比較，可以發現不管在收斂上與最後評估效果上，final 模型表現都好很多。

1. **未來展望**

首先，在 SE 的資料上，目前只有以站點間直線距離為參考，原先規劃是要測試使用google map 上的騎車路徑距離和時間去測試不同地理上的因素對站點存量預測是否有幫助，甚至超越直線距離。但是因時間問題這部分實驗的爬蟲程式還未完成，希望之後可以補上。

此外，希望可以加強視覺化與可解釋性的部分，例如將訓練完的模型權重在不同時間(TE)的變化視覺化，並和真實資料做比較，看看模型是否真的有學到在不同時間上站點間動態的供需變化。

最後我們希望可以讓模型只專注在需要的時段上，像是當前的任務是 Youbike，但基本上半夜的時候大約 5-6 小時腳踏車存量是幾乎不變的，雖然GMAN 模型有 TE 去幫助模型知道現在是一天中的哪個時間點，但是不代表這 5-6 小時對於白天的18-19小時的準確不會有影響。所以我們可以設計目標為白天的 18-19 小時，在輸入的部分分成是否給半夜 5-6 小時不變的資料去做比較。

1. **程式碼**

我們的程式碼與實驗資料都有開放到github 上了，網址如下:

[https://github.com/aaron1aaron2/NCCU\_110\_pytorch-and-ML\_Prediction-](https://github.com/aaron1aaron2/NCCU_110_pytorch-and-ML_Prediction-of-bike-shortage)

[of-bike-shortage](https://github.com/aaron1aaron2/NCCU_110_pytorch-and-ML_Prediction-of-bike-shortage)

1. **參考文獻**
2. Sensor distribution of the METR-LA and PEMS-BAY dataset.

檢自: <https://www.researchgate.net/figure/Sensor-distribution-of-the-METR-LA-and-PEMS-BAY-dataset_fig6_318316069>

1. PEMS-BAY benchmark

檢自: <https://paperswithcode.com/sota/traffic-prediction-on-pems-bay>

# Chuanpan Zheng, Xiaoliang Fan, Cheng Wang, and Jianzhong Qi. GMAN: A Graph Multi-Attention Network for Traffic Prediction, AAAI2020

# 檢自: <https://arxiv.org/abs/1911.08415>

# Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17). Curran Associates Inc., Red Hook, NY, USA, 6000–6010. 檢自: <https://arxiv.org/abs/1706.03762>

# A. Grover, J. Leskovec. node2vec: Scalable Feature Learning for Networks. ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), 2016. 檢自: <https://arxiv.org/abs/1607.00653>