# Wireless Multimedia Networks Lab1

# Throughput Prediction written by 313553024 蘇柏叡

---

## 0. 進行Fine-Tuning之前置作業與說明

在Lab 1中是使用虛擬器生成的資料集進行訓練，並且獲得預訓練的權重檔案。那在Lab 1中我們分別使用**Baseline model**、我們自行設計之模型進行訓練，並且於我們自行設計的模型中並應用於虛擬器生成的資料中獲得較佳的效果。此外，由於考量先前使用Normalized MSE Loss在訓練、測試集的表現上均低於了0.00002，因此我們將不再額外再新增全連接層或是殘差模塊，而是決定直接使用我們於Lab 1設計的模型架構針對真實的資料進行Fine-Tuning。此外，為了於後續章節討論需要凍結的Layer造成之影響(所謂凍結就是，該層Layer會保持原先的預訓練權重，並不會做更新)，因此我們會下章節針對我們先前於Lab1使用的模型架構進行**Recap**。

## 1. 針對在Lab1使用的模型進行Recap

首先在編碼器部分，其流程為將輸入維度為17的資料先經過一層線性層將其映射至256通道，接著經過兩個殘差模塊並且均保持256通道，最終進入一個線性層把256通道壓縮至128通道。接著，解碼器部分會將壓縮好的128通道送入兩層殘差模塊，並且保持128通道，最終在送入兩個線性層分別將128通道映射回256通道再壓縮回1維，並且通過ReLU將負值過濾為0後再進行輸出。至於殘差模塊的內部結構，我們會再於圖5、6進行更詳細的呈現與說明。

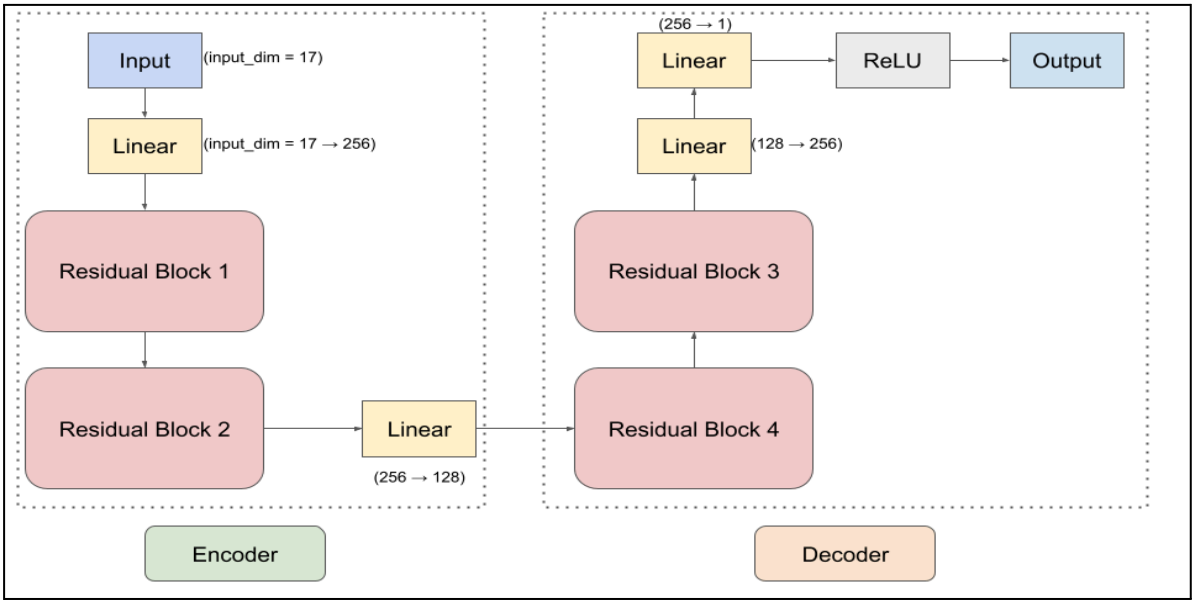


**圖1 為先前Lab 1自行提出模型架構圖**

在殘差模塊內部結構的部分，我們發現**由於編碼器中Residual Block 1和Residual Block 2結構相同，因此我們只使用圖2進行表示。**在此結構中分為兩個分支，在input x部分會是由先前線性層映射出的256通道，並且在殘差模塊內部會先經過一個線性層將通道壓縮成原先的一半，並且通過ReLU、Dropout(和原設定一致為0.2)後，再經過一層線性層，把通道映射回256通道，並且和原先Input x進行殘差連接並且輸出。在解碼器中使用的殘差模塊內部結構與先前**Residual Block 1和Residual Block 2結構相同，**只是Residual Block 3和Residual Block 4的input x為128通道，並且第一次通過線性層時是將通道砍半為64，再經過ReLU、Dropout後，再經過一次線性層映射回128通道再和input x做殘差連接並輸出。

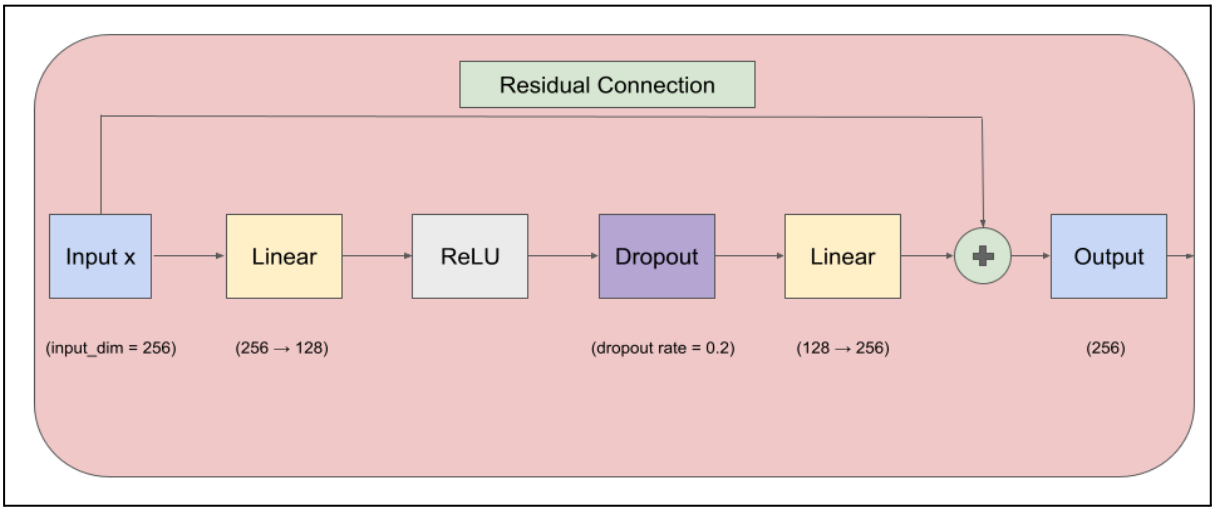


**圖2 編碼器中殘差模塊的內部架構圖**

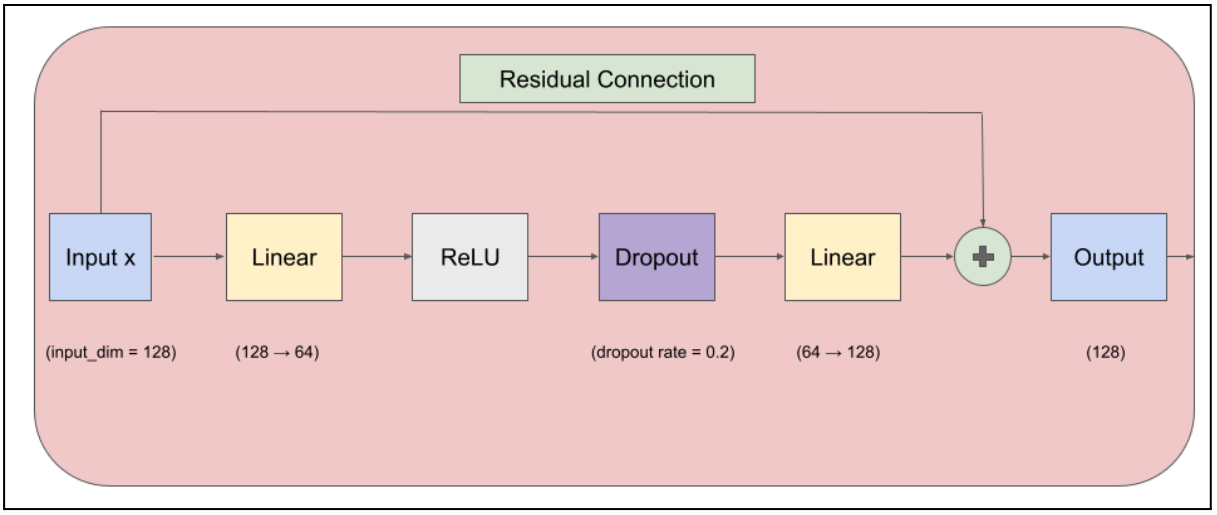


**圖3 解碼器中殘差模塊的內部架構圖**

## 2. Fine-Tuning過程

我們先從原本的Lab 1範例程式進行修改，在處理資料集時，由於第一個欄位是downlink throughput，因此我們會將第一個欄位視為目標類別，並只取後面的欄位。接著我們針對真實的資料集共9, 868筆資料進行切分，由於Lab 1時是採用9:1的比例進行切分訓練、測試。然而，若是採原先切分方式會導致測試集數量過少(測試集只會有987張)，因此我也會嘗試不同比例進行切割並比較，使其測試的資料盡可能不要產生嚴重稀缺。

至於使用原先在Lab 1訓練好的預訓練權重進行微調時需要注意以下情形。首先，為避免模型重新訓練，因此在微調時會選擇凍結部分層，並指會更新部分層的權重。另外在選擇Epoch數量的部分，因為有採取LR scheduler、並且加入Early Stopping，因此不必設置過小的Epoch，而是交由測試集的表現進而決定何時終止。至於學習率的部份，由於Fine-Tuning時不必From Scratch的去訓練整個模型，因此往往會採用比原先更小的學習率進行微調。故相較於原先設定lr為1E-4，我們在微調時使用了5E-5的學習率，使其可以達到較佳的微調之效果，至於優化器部分我們選擇的是Adam。

## 3. Fine-Tuning Performance:

### (1) 使用訓練、測試集為90%:10%並且針對編碼器中的第一個Residual Block進行凍結

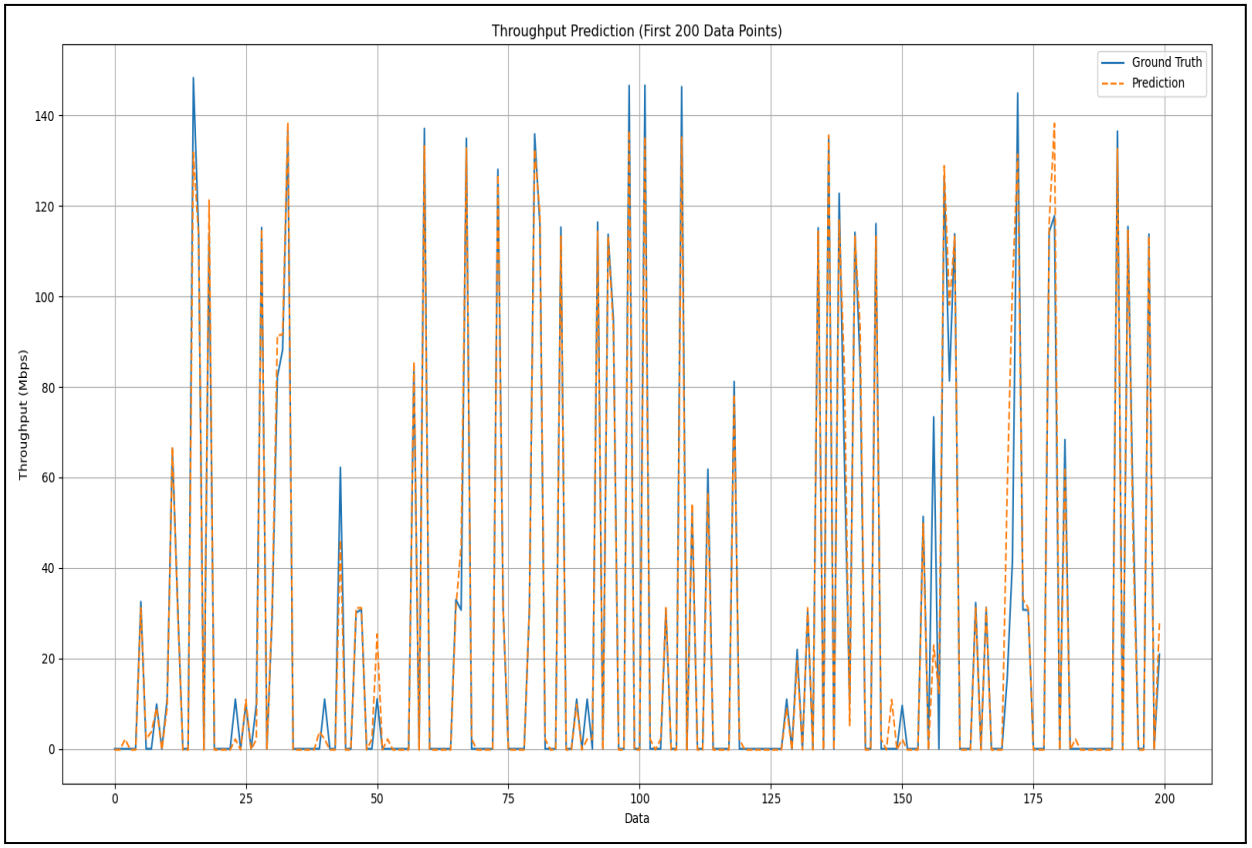


**圖4 為測試集前200筆的Throughput Prediction**

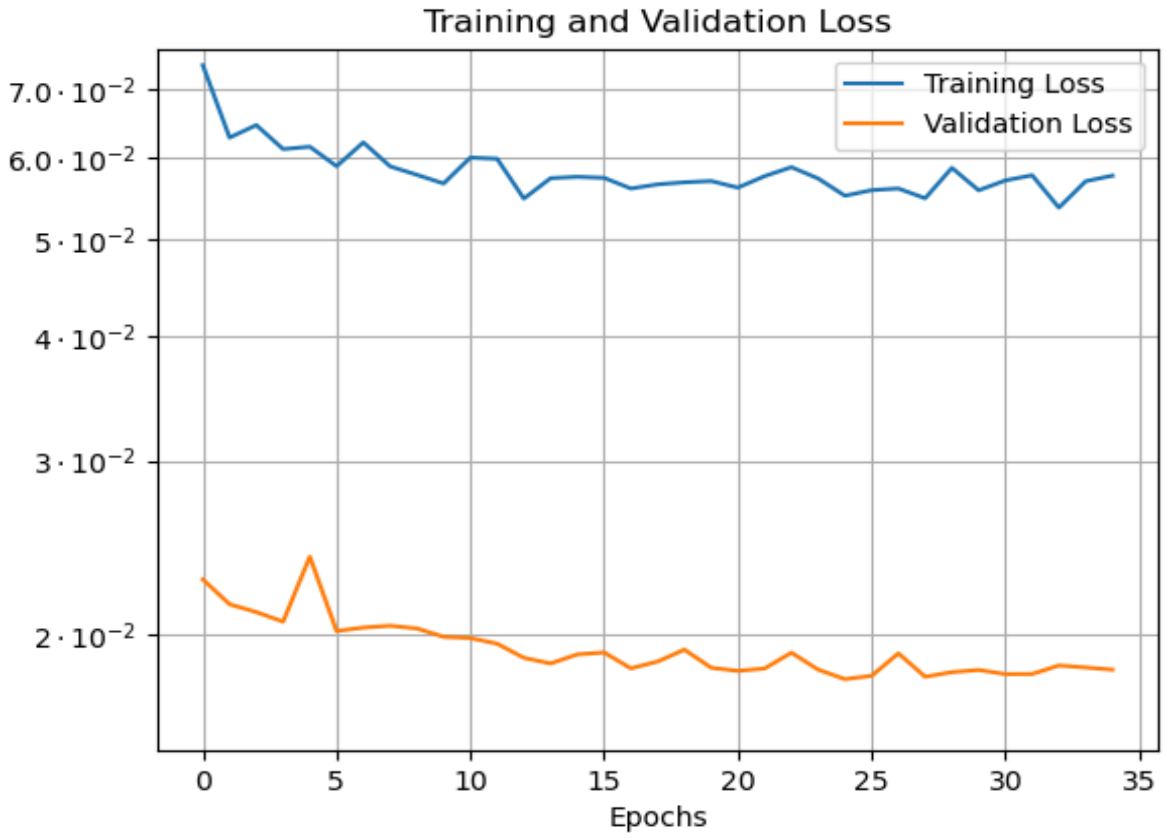


**圖5 為Fine-Tune過程中的訓練、測試損失函數圖**

在本組合中，我們發現在**GroundTruth為10**的情況下，模型預測的結果往往較不佳，在吞吐量較極端高的時候，大部分均能保持不落差過大的預測，僅有150-175之間的有點明顯落差。至於在損失函數部分，我們依然是採用**Normalized MSE Loss**，由於本組合是僅有987筆資料進行Fine-Tuning測試，故慘生訓練損失大約收斂至0.055，而驗證損失則是來到0.18的現象，因此我們推估此狀況應該是因為資料量過少造成的，因此接下來我們除了會嘗試凍結不同層進行比較之外，也會提高測試集比例並進行比較。

**(2) 使用訓練、測試集為90%:10%並且針對編碼器第一個Residual Block以及解碼器第一個Residual Block進行凍結**

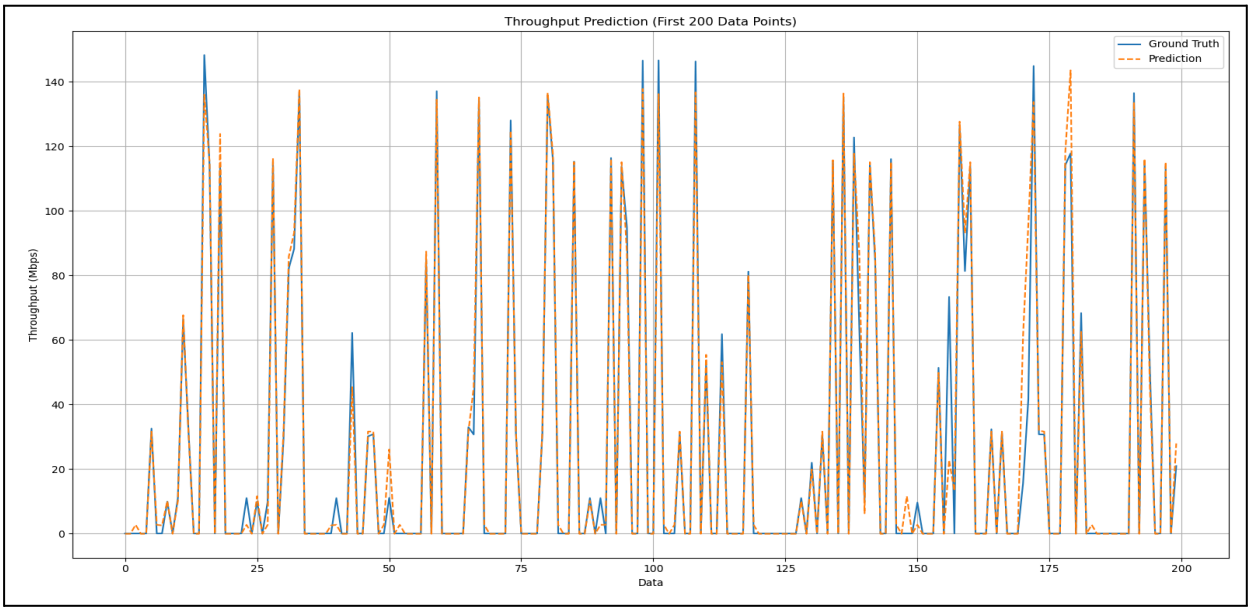


**圖6 為測試集前200筆的Throughput Prediction**

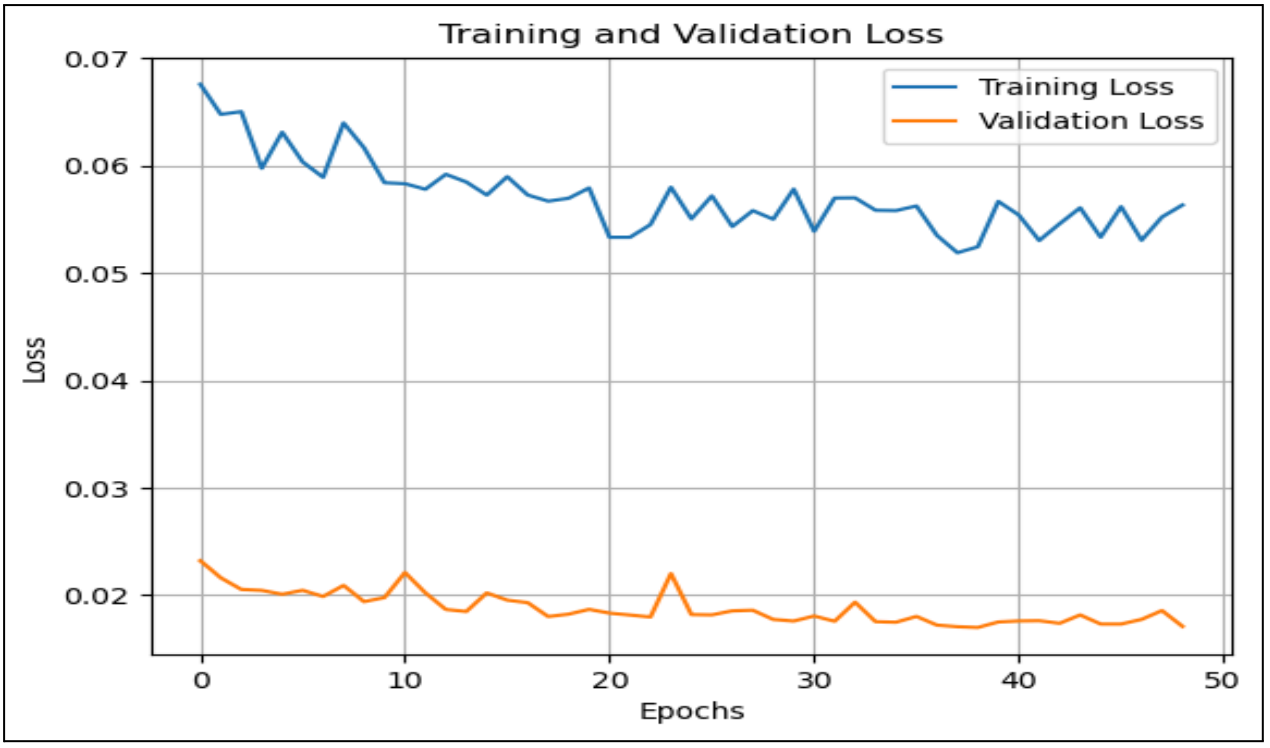


**圖7 為Fine-Tune過程中的訓練、測試損失函數圖**

在本組合中，相較於前一種組合新增凍結了解碼器的第一個殘差模塊，在吞吐量的預測上和第一種組合相比並無顯著地落差，至於在訓練與驗證損失部分則是發現訓練損失約為0.052，至於驗證損失則是約0.017，但誠如先前所述訓練資料量過於稀缺容易造成較不穩定的情況，因此我們在接下來會使用較大的測試集比例進行評估。

**(3) 使用訓練、測試集為75%:25%並且針對編碼器第一個Residual Block進行凍結**

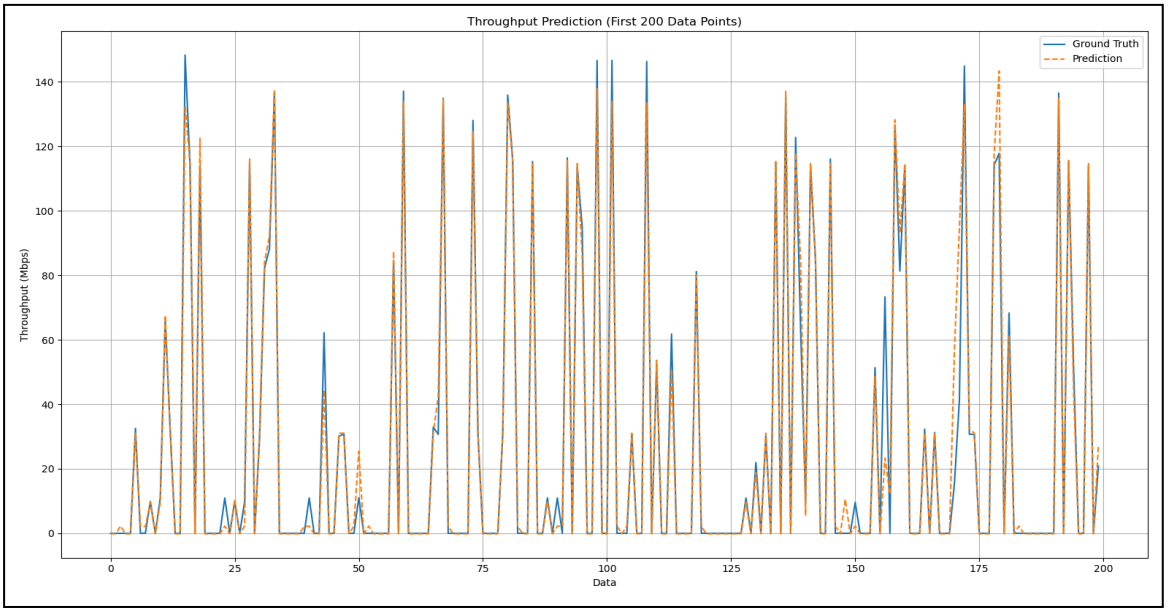


**圖8 為測試集前200筆的Throughput Prediction**

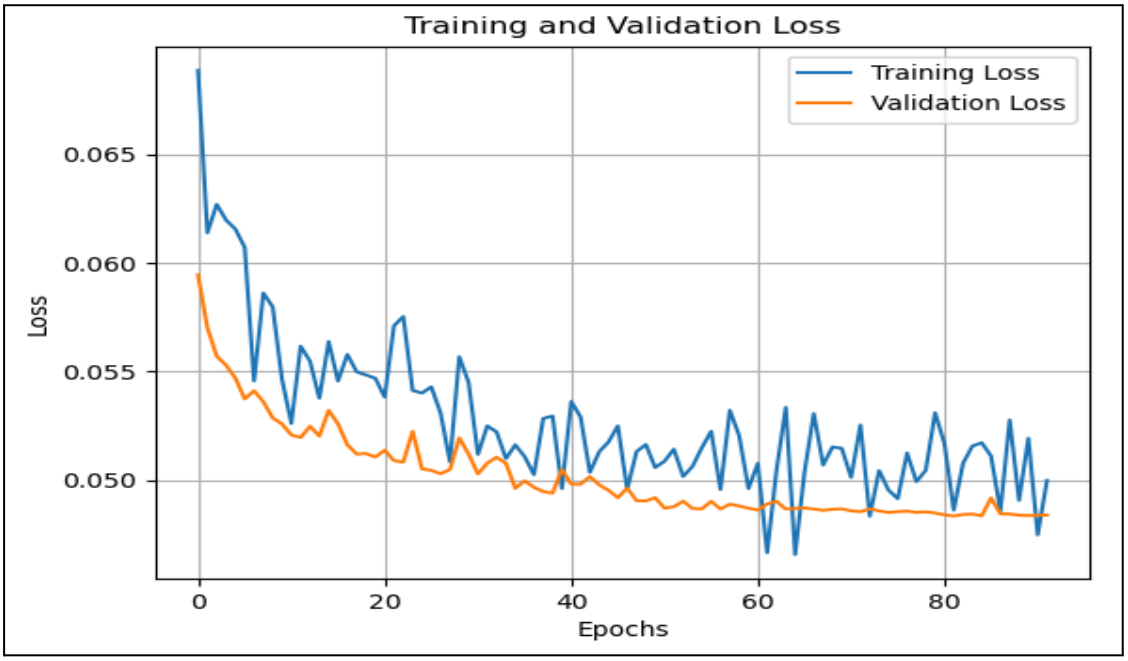


**圖9 為Fine-Tune過程中的訓練、測試損失函數圖**

在本組合中，我們將測試集的資料拉高到了近2500筆，可以發現此種切割方法，使訓練、驗證集的損失函數大約皆是有收斂至0.048以下，然而在訓練損失的部分較為震盪，反而是驗證集的收斂較為平穩。至於吞吐量預測的部分和先前的組合其實肉眼上看不太出明顯的落差，換言之除了先前提到的150-175及175-200區間存在了一些較為明顯的落差外其實第三種組合和先前預測的結果在肉眼觀測上並無極度偏差的情形。

**(4) 使用訓練、測試集為75%:25%並且針對編碼器第一個Residual Block以及解碼器第一個Residual Block進行凍結**

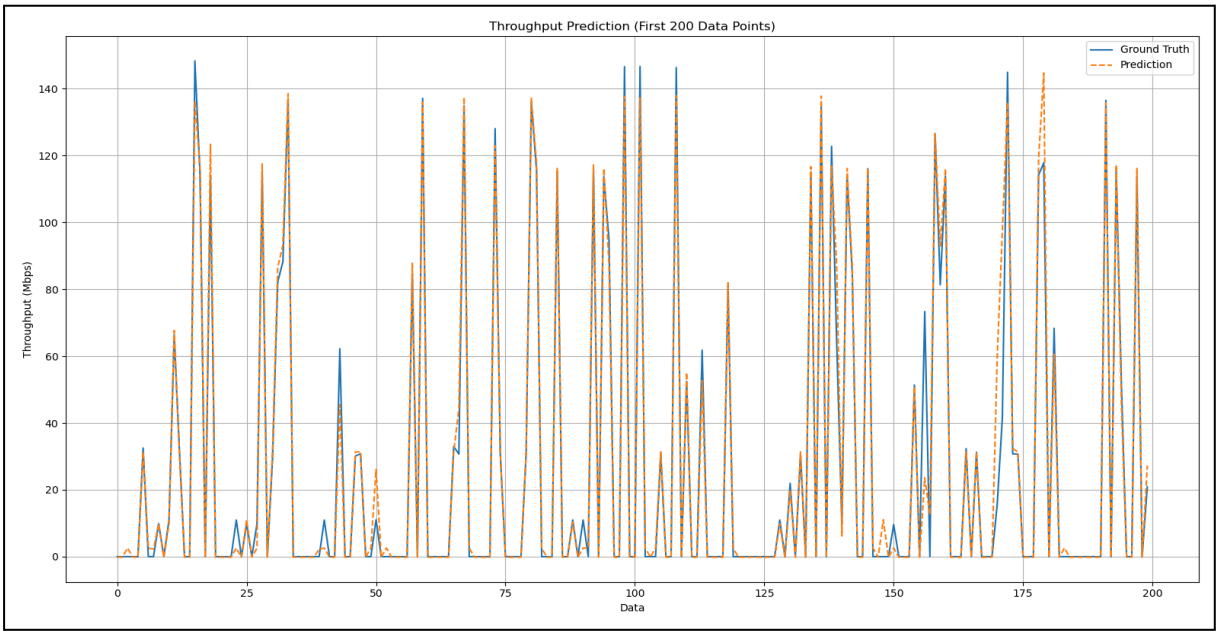


**圖10 為測試集前200筆的Throughput Prediction**

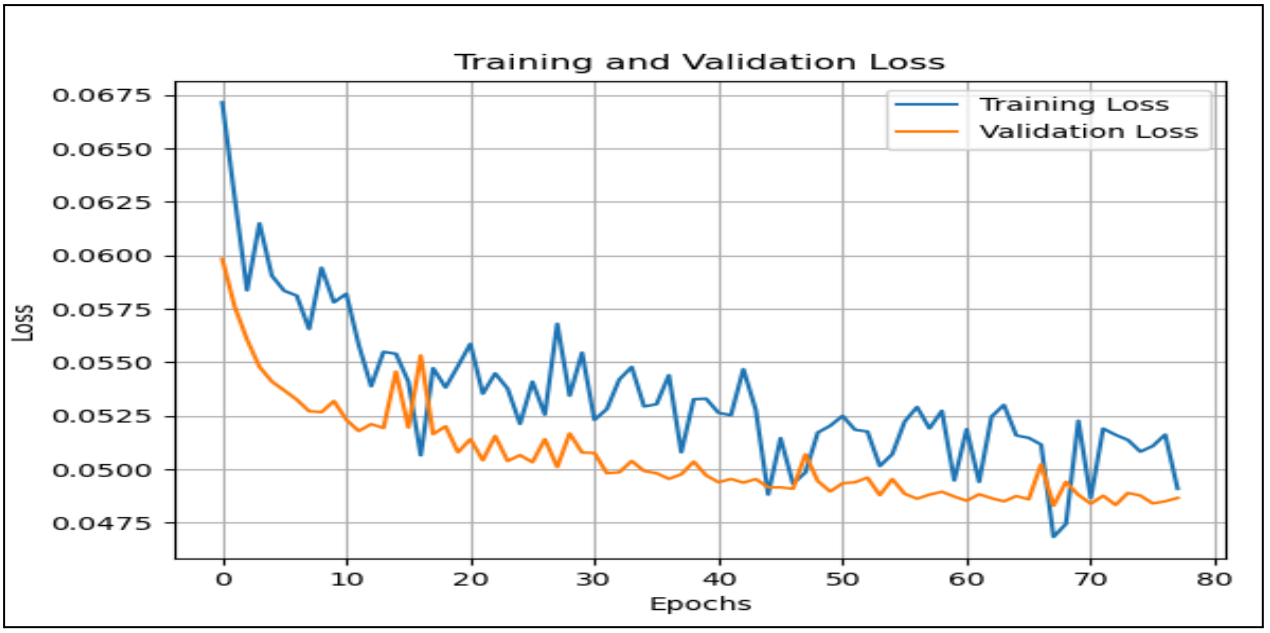


**圖11 為Fine-Tune過程中的訓練、測試損失函數圖**

在本組合中，在訓練、驗證集的損失函數如同第三種方法也是有收斂至0.048以下，然而相較於第三種方法，此種方法在訓練、驗證集上皆較為震盪。至於吞吐量預測的部分和先前的組合其實肉眼上看不太出明顯的落差。不過可以預期的是因為本方法在解碼器部分多保留了預訓練權重的一個殘差模塊，可以推測是因為原預訓練權重的資料分布和真實資料有較大的資料分布差異而造成了較為震盪的情況。

## 4. Conclusion

本次研究欲達成虛實整合，在Lab1中我們提出自行建構的模型，並且取得優異的表現，在Lab2中我們分別比較了不同資料分割以及凍結不同層的實驗，實驗結果表明，不論是只凍結了編碼器的第一個殘差模塊，或是多凍結了解碼器的殘差模塊，兩者在訓練、驗證損失與吞吐量預測上的表現其實差異不大。主要影響的因素仍然為資料分布與資料量的問題，由於真實資料相較於虛擬生成的資料不論是吞吐量抑或是資料量皆存在極端的落差，因此想利用虛擬資料並且結合真實資料達成虛實整合仍有不易。雖然在虛擬資料部分達到驗證集只會有0.00002的Normalized MSE Loss，但是本研究在使用其預訓練權重進行微調時，分別僅能獲得約0.02、0.048的Normalized MSE Loss，因此本人認為若是未來想達到更加的虛實整合效果，可以考慮收集大量資料集，並且進行訓練。另外，本人原先欲使用資料擴增方式針對真實資料進行合成，然而其效果反而比先前數種方法來的較差，因此並未放入討論。