GitHub link of my code:

https://github.com/jerrykao8787/mlFinal

model link of my code:

https://drive.google.com/file/d/1XQPkt8vlkBTyP7UgUuK3R-5-R-b4WAo_/view?usp=share_link

Reference if you used any code from other resources

參考這位作者的模型架構

https://www.kaggle.com/competitions/tabular-playground-series-aug-2022/discussion/349385

Brief introduction

在這次的作業中，我參考網路上的模型架構，利用 PyTorch 來實作深度神經網路，預測產品的故障率。

Methodology (Data pre-process, Model architecture, Hyperparameters, ...)

資料前處理:

因為 attribute_1 在 test 上出現 train 沒有的材料，所以把這個欄位的資料丟棄

另外嘗試丟棄 attribute_2、attribute_3，結果準確率下降(0.58612)

另外嘗試單獨丟棄 attribute_2，結果準確率下降(0.58864)

另外嘗試單獨丟棄 attribute_3，結果準確率下降(0.58899)

使用 pandas.get_dummies 把 attribute_0 編碼(one hot encoding, drop_first)

一開始我直接把缺的資料補零，但準確率很低(0.5409)

後來改用插補法補齊缺少的資料，假設不同筆測量資料彼此沒有關聯，因此選擇平均插補法，使用資料的平均值填補空白

另外嘗試使用 StandardScaler()把原始資料縮放，結果準確率下降(0.56172)

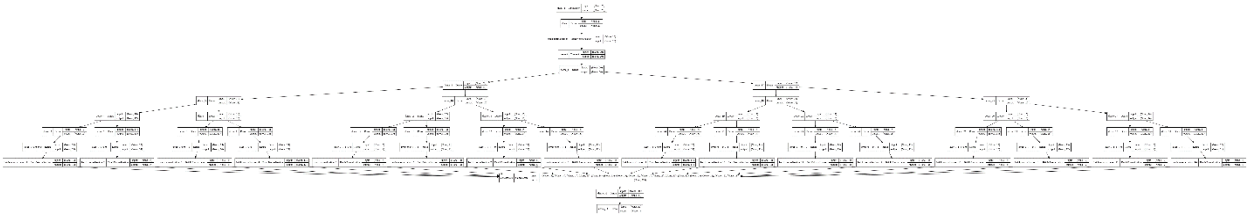
使用 sklearn.model_selection.train_test_split 將 10%的訓練資料切分給 valid

另外嘗試使用 20%的訓練資料切分給 valid，結果準確率下降(0.58687)

模型架構:

使用全連接神經網路，結合 ReLU 激勵函數，使用 BatchNorm1d 讓訓練過程中每一層神經網絡的輸入保持相同分佈，搭配 Dropout 正則化方法來對抗過擬合。架構的部分參考自網路。

(下面圖片有點不清楚，高清版本: https://www.kaggle.com/competitions/tabular-playground-series-aug-2022/discussion/349385)



由於這次的預測結果只有 0 和 1，因此使用 BinaryCrossEntropy 損失函數，同時在模型最後一層使用 Sigmoid 激勵函數，產生 0 到 1 的值，預測產品故障率。

超參數

learning_rate = 0.001

optimizer = Adam

scheduler = ReduceLROnPlateau (若驗證資料集的損失沒有下降，就調低學習率)

epoch = 300

因為 colab 給的記憶體夠大，因此設定 batch_size 為 len(整份資料集)

另外嘗試使用 batch_size = 100，結果準確率下降(0.58578)

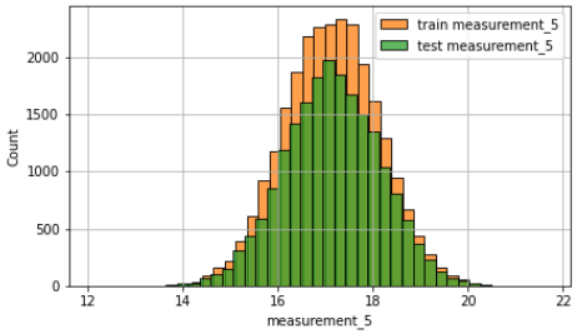
訓練過程

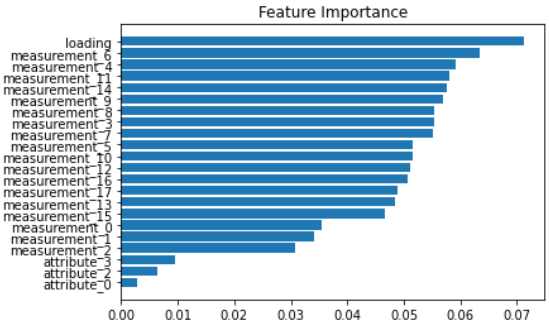
基本上深度學習模型訓練過程的 code 都差不多，主要是 optimizer.zero_grad()、model(inputs)、算 loss、loss.backward()、optimizer.step()，這邊就不贅述。

訓練時，儲存驗證資料集損失最低的模型版本，用於測試資料中。

Summary

透過這次作業，讓我學到深度神經網路的應用，利用機器學習來解決日常生活遇到的問題。這次的訓練資料跟以往作業很不一樣，很難找到一個切分點來直接區分產品故障與否。如果直接把訓練資料和測試資料視覺化，幾乎每個維度的結果都像下面這樣，訓練和測試資料集的分佈很類似，平均、標準差也差不多，因此透過深度學習來幫助我們找出資料的特色、相異處，是最輕鬆省力的方式，還可以藉此了解影響產品故障的因素。





原先我打算用 HW3 學到的 Decision Tree、AdaBooest、Random Forest 來訓練，但是因為這次作業需要輸出浮點數的機率值，而不是直接預測正常或失敗(0 或 1)，因此 HW3 的方法沒辦法直接套用在這次作業中。

Result