

# Introduction to Machine Learning 2024

## Defect Classifications of AOI

Group 5 陳昱仲 邱泓斌

## Introduction

在這次的 Final Project 中，我們參考 Alexnet 的架構實作了一個分類模型，並且利用許多技巧加速訓練的速度以及模型的準確度，最終在比賽當中我們成功拿到 98.9% 的成績。

## Method

因為訓練資料並不像 ImageNet 那樣龐大，因此我們選擇縮小模型大小，最終使用了三層 CNN + MaxPool 以及三層 Dropout + Linear，模型總共大小約為 60 MB，並且使用 Cross Entropy Loss 以及 Adam Optimizer 來訓練模型。我們總共訓練 150 個 Epochs，設定 batch\_size 為 128。在一開始測試的時候，我們選擇 8:2 把資料集切割成 Training Dataset 和 Testing Dataset，不過後來發現原始的資料數量只有 2528 張，切成 8:2 會導致大量資料沒有被拿來訓練，因此最終使用了 99:1 的比例切割資料。

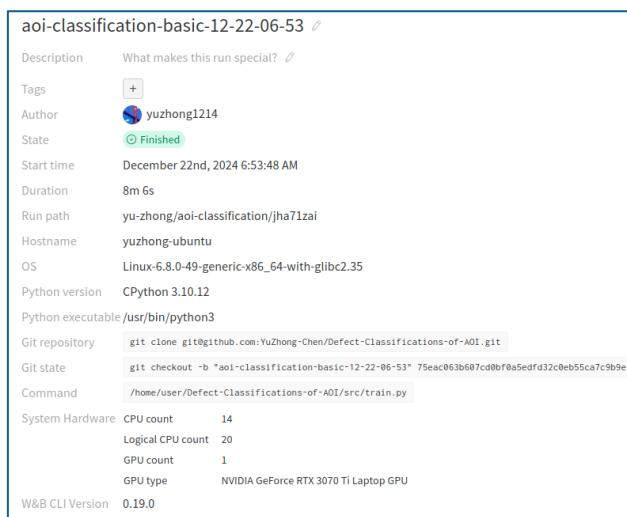
4	Layer (type)	Output Shape	Param #
5			
6	=====	=====	=====
7	Conv2d-1	[ -1, 64, 85, 85]	3,200
8	ReLU-2	[ -1, 64, 85, 85]	0
9	MaxPool2d-3	[ -1, 64, 42, 42]	0
10	Conv2d-4	[ -1, 128, 21, 21]	204,928
11	ReLU-5	[ -1, 128, 21, 21]	0
12	MaxPool2d-6	[ -1, 128, 10, 10]	0
13	Conv2d-7	[ -1, 256, 10, 10]	295,168
14	ReLU-8	[ -1, 256, 10, 10]	0
15	MaxPool2d-9	[ -1, 256, 4, 4]	0
16	AdaptiveAvgPool2d-10	[ -1, 256, 4, 4]	0
17	Dropout-11	[ -1, 4096]	0
18	Linear-12	[ -1, 2048]	8,390,656
19	ReLU-13	[ -1, 2048]	0
20	Dropout-14	[ -1, 2048]	0
21	Linear-15	[ -1, 2048]	4,196,352
22	ReLU-16	[ -1, 2048]	0
23	Linear-17	[ -1, 6]	12,294
24	=====	=====	=====
25	Total params:	13,102,598	
26	Trainable params:	13,102,598	
27	Non-trainable params:	0	
28	-----	-----	-----
29	Input size (MB):	0.25	
30	Forward/backward pass size (MB):	9.44	
31	Params size (MB):	49.98	
32	Estimated Total Size (MB):	59.67	
33	-----	-----	-----

# Tricks

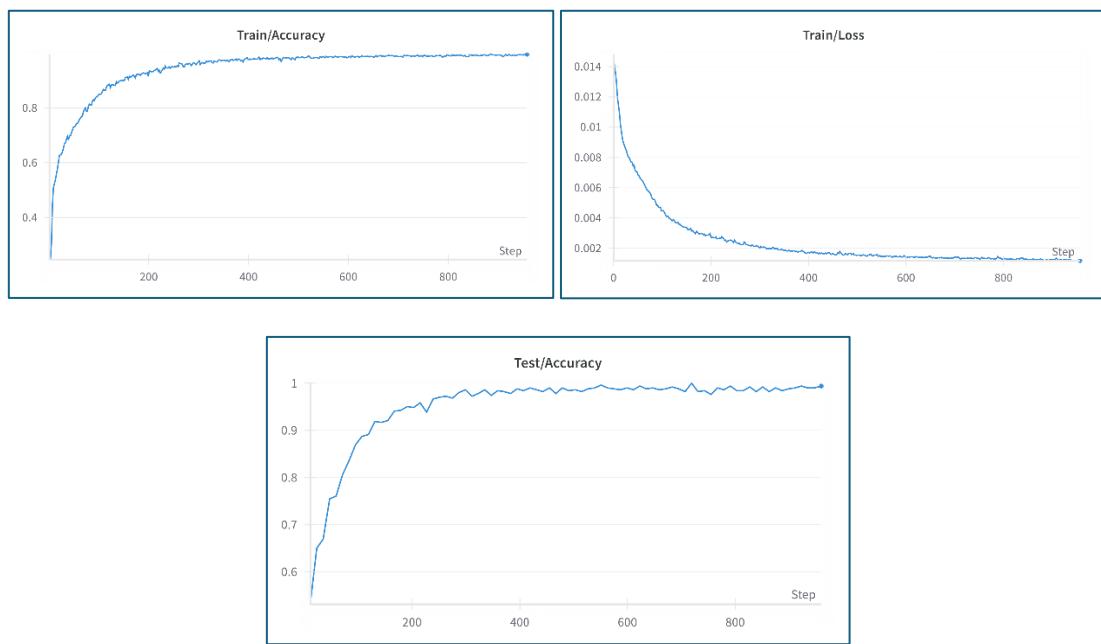
這邊介紹三個我們使用到比較重要的 **Tricks**。

## 1. Weights & Biases

因為我們的資料集/模型都很小，因此參數的選擇將會大幅度影響最後的準確度，為了有效率的追蹤訓練結果，我們選擇使用 W&B 都做 **Logger**。此工具不但會保留過去所有的訓練結果，方便我們交叉比對結果，也會記錄下當時的 **Git State**，幫助我們輕易的重現當時的實驗數據，同時它也會記錄下所有的參數、訓練環境等等資訊，大幅度減輕我們 **Fine-tuning** 的難易度。



▲ 此工具記錄下的環境資訊



▲ 訓練結果

## 2. NVIDIA Nsight Systems

一開始訓練一次模型就要花費我們二十到三十分鐘，考量到此資料集以及模型大小，我們覺得相當不合理，因此嘗試使用 Nsight Systems 分析我們的程式。這個由 NVIDIA 推出的 Profiler 可以幫助我們快速找到整體 Pipeline 的 Bottleneck，而且不同於 PyTorch 內建提供的 Profiler，此工具還可以得知 CPU 與 GPU 的使用情況以及兩者互動的情形。藉由此工具，我們找到問題的所在: IO 端。因為使用的 Dataset 算是自定義的，因此我們參考 PyTorch 的教學建立 Dataloader，然而在這個教學當中，他們預設在每次使用資料時，重新從 Disk 讀取資料到 Memory，再搬移到 GPU 上面進行 Training，這個流程耗費了大量的時間，並且因為不能平行化所以拖累了整個 Pipeline，考量到整體 Dataset 的大小大約只有 0.6 GB，而我們使用的 GPU 是 NV 3070 Ti，總共有 8GB 的 RAM，所以就直接把 Dataset 放到 GPU 當中，大幅度減少資料讀取時間，最終訓練一次模型只要八分鐘左右。

## 3. Torchvision

因為資料集的數量很少 (總共 2528 張)，而且還要切割成 Training Dataset 和 Testing Dataset，導致真正可以拿來訓練的資料數量嚴重不足。根據我們的觀察，我們發現資料並不像是日常照片，它不並具有方向性，因此同一張照片只要水平或垂直翻轉就可以直接當作一個新的資料，藉由設定這兩種翻轉各 50% 的機率，以及在  $\pm 10$  度內旋轉，我們就可以獲得更多資料，最終在利用 Normalized 統一資料的 mean/std，讓我們把模型準確度從 94% 提升到 98% 左右。

## Future Direction

因為時間有限，我們沒辦法確實的 Fine-tuning 所有參數以及模型架構的選擇，未來除了可以繼續調整來增加模型準確度，也可以考慮換成其他模型，像是當前最強的視覺模型架構 Vision Transformer (ViT)，我們預期這些 Future Work 可以進一步的加強模型準確度。

## Conclusion

在這次的 Final Project 當中，我們學會並運用了許多技巧，像是從零開始建構一個 Classification Model、利用諸多工具以及技巧幫助我們分析並加速模型訓練過程及結果，非常感謝教授以及助教們的指導與幫助。