Learning With Machine

文 / B09 楊泰萱

我的專題領域是用機器學習 (machine learning) 去加速模擬純量場的取樣 (sampling) 過程，基本上會需要有統計物理的基礎知識以及如何寫程式就可以進行我的專題研究，但要有一些更進一步的改良，額外的物理及資工領域的知識會有所幫助。例如，為了讓程式有更快的 runtime，對複雜度理論有一點概念會更可以知道要如何提升機器學習算法的效率；對 pytorch 及 Linux 的操作有先備知識可以提升 coding 的效率；對 gauge theory 以及 symmetry 熟悉的話，也可將原先純量場的框架修改成對 gauge theory 的框架。但整體而言是偏向於資工領域應用於物理的專題。

由於專題的主題主要是陳俊瑋教授對學長之前的工作進行改良，所以一開始在文獻探討方面較為輕鬆，基本上就是學長跟我們講解原理後我們就開始進行 coding 了，但改良的過程才是真正困難的部分。我們原先的算法被荷蘭 Amsterdam 的 group (下簡稱 A group) 加速了約一個數量級，所以當我們把原先學長的 code 理解之後，又要去看 A group 的 paper 並且進行比較，然後為了對 A group 的 paper 進一步優化，我們開始查找大量的文獻，並同時複製出 A group 的 code。在我們複製出他們的結果後，我們也發現了該模型不足的地方，並著手進行優化，但是我們所使用的 V100 GPU 在當時因為送修而停工了兩個月，也因此耽誤到了研究的進度。在 V100 GPU 修復完後，我們開始實踐我們的想法，大約在一個半月後，的確得到了不錯的結果，不過在後來與 A group 開會的過程發現他們也用了相同的 idea 並且還有對另外的地方進行優化，所以效率仍高於我們的結果並且已經寫好文章準備上 arxiv，所以我們最後的結果並沒有成功發表。

時間線整理

2021 年 7 月底 - 2021 年 9 月初

理解機器學習算法背後的原理。

2021 年 9 月初 - 2021 年 10 月中

開始看學長和先前論文的 code 並在自己電腦上實作。

2021 年 10 月中

在 meeting 中得知 A group 得到了快出一個量級的加速。

2021 年 10 月底 - 2021 年 12 月初

複製出 A group 的 code 並進行測試。

2021 年 12 月初 - 2022 年 1 月中

根據 A group 的算法尋找出更優化的算法並實作。

2022 年 1 月中 - 2022 年 3 月底

V100 GPU 維修。

2022 年 3 月底 - 2022 年 5 月初

成功實作出優於 A group 的算法。

2022 年 5 月初

在與 A group 的 meeting 中知道他們又有更快的算法並與我們的 idea 高度重疊，且他們還有做額外的優化，並且已經幾乎完成論文的撰寫。

2022 年 5 月中 - 2022 年 6 月底

提出新的 idea（不同於 A group）並實作，得到與 A group 新算法差不多的速度。

2022 年 7 月初

A group 的新論文放上 arxiv，我們也結束此專題。

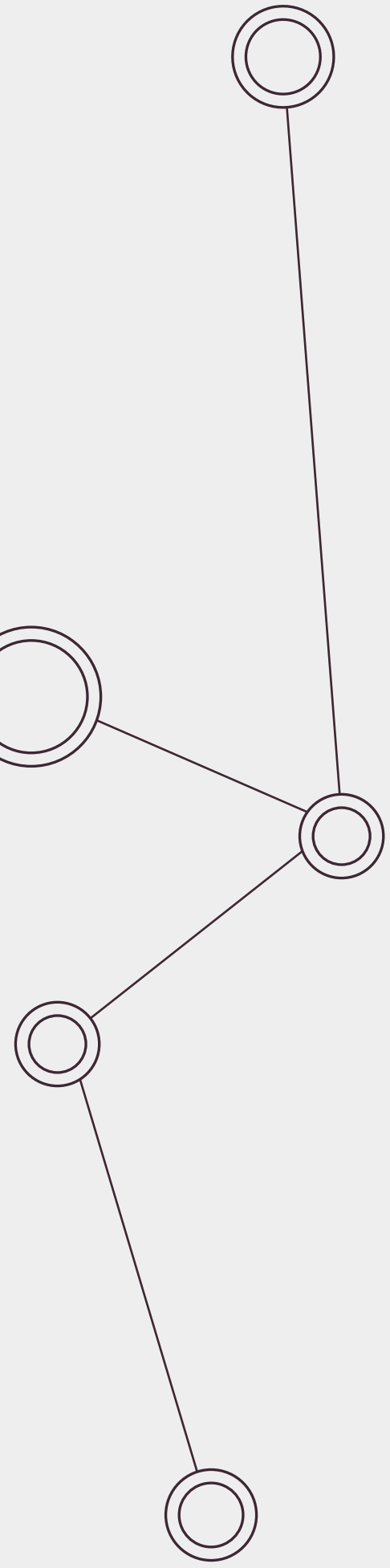
我個人覺得這份專題很像是在做實驗，只是實驗地點不是在實驗室而是自己的電腦 (或是學校的伺服器)，我們會試不同的想法，把 code 寫出來，然後放著跑一個晚上隔天早上起來看實驗結果，有時也要整理實驗結果畫圖表，來看出我們到底改進了多少。在文獻閱讀的部分也是頗為相似，文獻上的理論都不會很難，基本上一天可以看好幾篇論文，但是要能實作出論文的程式卻是一件非常耗時的事情，很像在基礎物理實驗或是近代物理實驗課中，你知道實驗該怎麼做，步驟基本上也寫好了 (但論文不會那麼詳細)，但是實際動手下去做實驗仍需要很長一段時間。反之，讀理論的論文，我們經常需要花大半時間釐清論文所跳過的步驟以及論文引用的結論是怎麼被推導出來的 (又要看另一篇論文)，光是要完全看懂一篇論文可能就是一個禮拜的事情。

在所需能力的細節方面，這份專題中最重要的部分反而是 coding 以及改良神經網路架構的能力而非物理，所以才會說這像是一份以資工領域為主的專題。例如專題的重點是如何快速地將已經發展完善的神經網路架構套用在我們要模擬的系統上、如何改良神經網路架構將物理系統的特性寫入其中，降低神經網路的參數使其更有效率，在 GPU 有限的運算資源及記憶體下跑出最高效的模擬結果。

從這次經驗中，我感覺到資工領域優化這塊領域，效率是非常非常的重要，大部分你能想到的 idea 其他的 group 也想的到，所以會變成誰先把 code 實作出來誰就能先發表文章。A group 原先的工作也是將其他的神經網路架構用在我們要模擬的系統上，並且這個架構可以根據要模擬的物理系統 encode 系統本身的 symmetry (Equivariant neural network)，進而減少了許多神經網路的參數，因此取得了巨大的效率提升。

基本上這類改善效率導向的研究可以說是有利有弊，好處是上手起來較為簡單，入門的難度沒有那麼高，如本文一開始所述，只要有能力寫出機器學習的模型以及一些統計物理的基礎知識即可上手，且找到一些方法進行優化並非是一件困難的事情，成功的機率算是蠻高的。但是壞處就是有可能會遇到跟我一樣被另外的 group 搶先實作出自身想法的經驗，由於我組員和我都是大二生，會面臨到課業的問題，期中和期末各會有兩周基本上沒辦法處理專題，並且在我們先前都沒有實際從 0 到 1 刻出 machine learning model 的經驗，我個人的程式能力也有待加強，且能衝刺專題的寒假還遇到了兩個月的 GPU 停機，從個人能力到 GPU 系統上都導致了一定程度的延遲才造成 idea 被 A group 先實作出來的結果。

雖然沒有成功發布，但我也學到了許多經驗以及知識，除了上述對效率的感想外，我也藉由專題更加了解 simulation 領域的樣貌，也知道自己對這個領域是真的有興趣。最後，再次感謝陳俊瑋教授的指導，以及我的組員賀崇恩，使我學到了各種不同的神經網路架構、如何從無到有刻出一個機器學習模型、如何使用 Linux、以及如何利用物理的 sense 去優化整個模擬過程。



附錄：專題簡介

Task：

從以下的機率分布中進行採樣：

$$P[\phi(x)] = \frac{e^{-\int(\nabla\phi(x))^2+s\phi(x)^2+\lambda\phi(x)^4}}{Z}$$

其中 Z 為 normalization constant、$\phi(x)$ 為場的 configuration，s 和 λ 為常數。

Model：

由於我們可在高斯分布中取樣，

$$P_0[\phi_0(x)] = e^{-\frac{1}{2}\int d^n x\phi_0(x)^2}$$

Flow model 的想法就是從一個簡單的高斯分布為出發點，將一函數 f 作用於原先取樣之，得到一新的 conf-iguration：

$$\phi(x) = f(\phi_0(x))$$

由於所有 configuration 出現的機率之總和需為 1：

$$P_{model}(\phi(x)) = P_0(\phi_0(x))det(\frac{\partial\phi(x)}{\partial\phi_0(x)})^{-1}$$

所以我們可以利用 f 進行變數變換來得到我們所想要的機率分布。Loss function 為 KL divergence，可以用來衡量兩機率分布的不同：

$$Loss = \int d\phi(x)P_{model}(\phi(x))log(\frac{P_{model}(\phi(x))}{P(\phi(x))})$$

$$= E_{P_{model}(\phi(x))}log(\frac{P_{model}(\phi(x))}{P(\phi(x))})$$

E 表示 expectation value，實作上來說就是我們在 $P_{model}(\phi(x))$ 中進行取樣，然後計算 $log(\frac{P_{model}(\phi(x))}{P(\phi(x))})$。那要如何在 $P_{model}(\phi(x))$ 下進行取樣呢？其實很簡單，就是我們在高斯分布中取樣出的 $\phi_0(x)$，經過函數 f 後，所得到的 $\phi(x)$，雖然我們並不知道 $P(\phi(x))$ 中的常數項 Z，但由於取 log 之後常數項會移出去，所以可以忽略。

而 Neural Network 所扮演的角色即是函數 f，可是 f 不能隨意亂選，由於對一任意 n*n 的矩陣 determinant 的計算，其複雜度為 $O(n^3)$，其運算時間過長，故我們要尋找一些函數 f 仍保有 generality，且其 determinant 的計算的複雜度為 O(n)，其中有 realNVP (學長 previous work)、Neural ODE flow (A group 的 work)、Neural spline flow (另一個 group) 等。此部分為專題著墨的核心，有興趣的同學可以去細查這些 flow model 的架構。

這種取樣方法的好處就是，我們只要有初始狀態，經過一個 flow 即可生成我們所想要的 configuration $\phi(x)$，比起用經典的 Hamiltonian Monte Carlo 方法，一步僅能生成一組 configuration (Markov chain) 要來的更有效率。

Note：

在程式實作上，我們會將積分轉換成在格點的上的 summation：

$$\int \to \sum$$

而 summation 的範圍為我們格點的大小。