2020暑期實習研究報告

李俊逸

# 緒論

* 前言

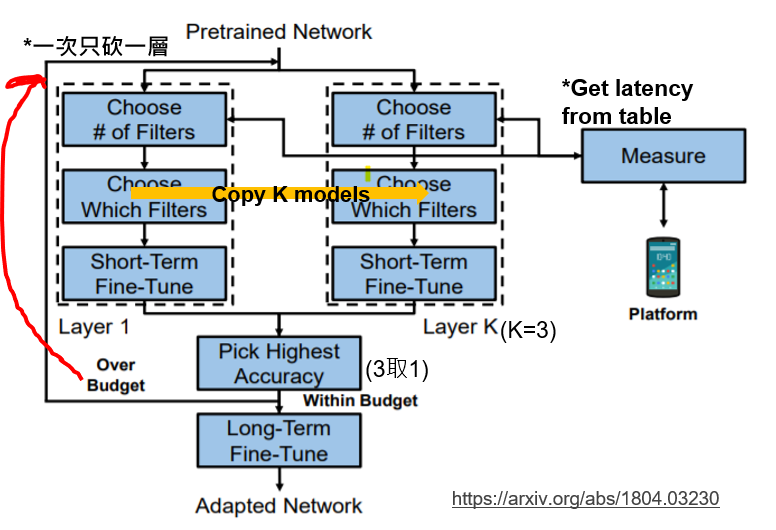
本次實習計畫透過實做NetAdapt論文, 實現model compression之目的。

Model compression由剪枝演算法(pruning)實現, 透過刪減模型中的參數減少模型計算量, 進而提升運算速度，同時維持模型之準確度(accuracy)。具體的做法是，多次刪除各層中較不重要的參數，每刪除一次就對模型進行再訓練(以維持模型準確度)，刪到符合限制(constraint)為止。舉例來說，若希望模型加快兩倍，則應刪除參數直到模型的延遲為原本的一半為止。

NetAdapt是一種剪枝演算法, 其主打使用direct metric, 換句話說, NetAdapt取得model計算延遲(latency)的方法是直接寫模擬模型的程式(tflite)在實際裝置上, 測量不同情況下, model內每層卷積層(convolution layer)的延遲。

本次實習計畫使用PyTorch1.2.0版本做為framework，以ResNet-18模型，測量剪枝前後在ImageNet-2012上的表現，目標為在指定模型上實現NetAdapt演算法機制。

* 演算法簡介

右圖是NetAdapt演算法的流程。給定一個pretrained network，接著根據有K層可以被砍，複製K個model進行剪枝，一次剪枝後，會從K個新砍的model中挑出準確度最好的model，接著判斷是否符合給定的budget，若符合則進行長期訓練，否則依照右圖紅線往回走，重複以上動作直到達到budget為止。另外，latency由實際在裝置上跑程式，測量不同filter(i.e.參數)數量下的latency，並生成table，使pruning時能夠直接查表得知latency。

* 實現過程

我將GitHub上的一位碩士生(NatGr)所做的NetAdapt Implementation (on WideResNet-40-2) 改寫，嘗試應用在ResNet-18上。這份專案中共有幾項重要的功能: 生成網路以及能夠更改網路結構、選擇多少參數以及哪些參數要被砍、資料前處理、以及做剪枝演算法的主要程式(main)，由於NatGr的code只專門應付WideResNet，故有諸多之處需要修改，以符合新模型的架構，需要更改之處主要是更改網路結構以及砍多少、砍哪些參數的函式。

最後實現的結果是，能夠對ResNet-18中的其中8層做剪枝，不過這8層不能被砍到消失，否則模型的forward function無法運行(基於時間限制，沒辦法implement出來)，其他的功能大致上與NatGr的code相同，也就是根據NetAdapt演算法以及調整超參數(hyperparameter)，達到模型剪枝的效果。

# 研究方法

我進行了五次的實驗，調整不同超參數下，檢視模型剪枝成效，以下將介紹結果最佳的實驗之各項實驗設定。

* 實驗設定

模型最終要減少30%的latency。

訓練資料分為短期及長期，短期指的是刪除一點點參數之後進行的訓練，長期則是剪枝結束後進行的訓練。在此實驗中，短期使用1個training dataset，而長期則使用50個進行訓練。

每次while loop大約減少1.5%的latency，每多一次iteration會降低pruning的量。舉例來說，第一次pruning減少1.5%，第二次減少1.5% \* 0.98，第三次減少1.5% \* 0.98 \* 0.98，依此類推。

可剪枝的層數為8層(ResNet-18有18層)，事實上最大的剪枝數量可以提升到18層，不過由於ResNet的結構有shortcut，剪了一層就需要再修改其他層，實現有其複雜度，故先實現較簡單的8層。

每次prune一層layer時，選擇不重要參數的方法是：將每個filter的參數做L2-Norm運算，將結果由高到低排列，越低代表越不重要，就會先被砍掉。

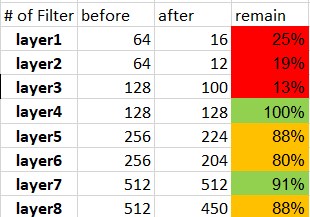
# 結果分析

* Overall result

Pretrained ResNet18 Top 1 Accuracy: 69.14% -> 67.16% (↓1.98%)

Pretrained ResNet18 Top 5 Accuracy: 88.83% -> 87.49% (↓1.34%)

Pretrained ResNet18 Latency: 0.33sec/image -> 0.23sec/image (↓0.1sec, 30% latency)

* Detailed result

前三層最終被砍的較為嚴重，後五層較無損傷，

不過這已經是最好的結果，原因是每一次模型要被

剪枝時，都會依據最有效率的剪枝做法(掉多少準確度/

減少多少時間)選出最好的模型。所以可以合理推測，

後面的convolution layer不能被砍太多，否則會造成

Accuracy掉太多的狀況。

# 結論

在暑期期間，我藉由改寫Github上的程式碼，使NetAdapt演算法能夠應用在ResNet-18

上，並且能夠在減少小於2%的accuracy之下，達成減少30% latency的效果。

不過若要實現不同model上能運行NetAdapt演算法，需要對程式碼及網路架構有深度了

解，此外需要修改大量的程式碼。基於時間問題，我在實作僅實現了能夠砍8層，而不能讓

18層中每層都能夠被砍，若能將所有層數剪枝程式碼實現，效果會更佳。