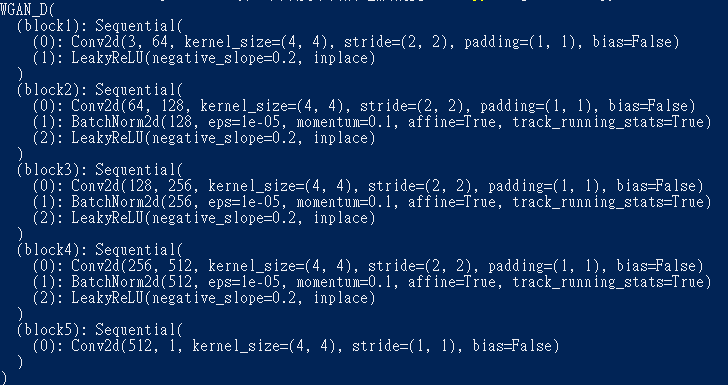
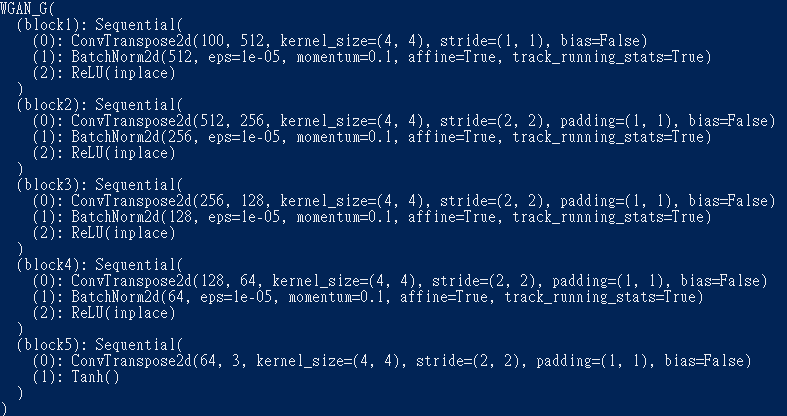
深度學習於電腦視覺 作業三

資工所碩一 r07922103 李俊賢

Problem 1：GAN

1. Describe the architecture & implementation details of your model.



在第一題中我實作了以convolution架構的WGAN。

WGAN\_G代表WGAN的Generator，負責把輸入的(batch\_size, 100, 1, 1)轉化成(batch\_size, 3, 64, 64)的一堆假圖片。WGAN\_G的實作細節如上圖所示，利用convTranspose2d把100 channel變成512 channel再從512 -> 256 -> 128 -> 64 -> 3 channel。

WGAN\_D代表WGAN的Discriminator，負責把輸入的假圖片(batch\_size, 3, 64, 64)轉化成(batch\_size, 1)的格式，其中1是用來判斷圖片的真假。實作細節如上圖，利用conv從3 -> 64 -> 128 -> 256 -> 512 -> 1 channel。

1. Plot 32 random images generated from your model.



1. Discuss what you’ve observed and learned from implementing GAN.

我認為我學到三件事。

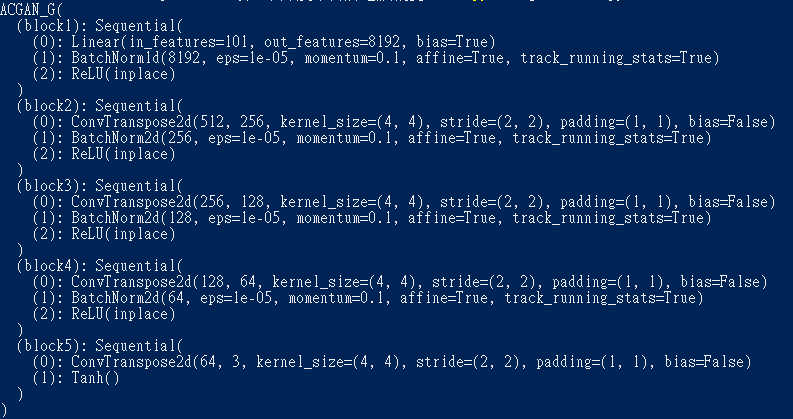
一：GAN真是很難Train，他的loss很容易不穩定起起伏伏，也很容易mode-collapse，其loss數值也沒有實質上的距離意義。因為他太不穩定，所以我採用比較好train的WGAN來實作，WGAN的精神就是把KL divergence換成Wasserstein距離，可以更穩定而且能使loss的數值有意義。所以整體做起來會更有成就感也更有實作的感覺，也因為數據都是有意義的，也可以從中看到他們學習的變化。

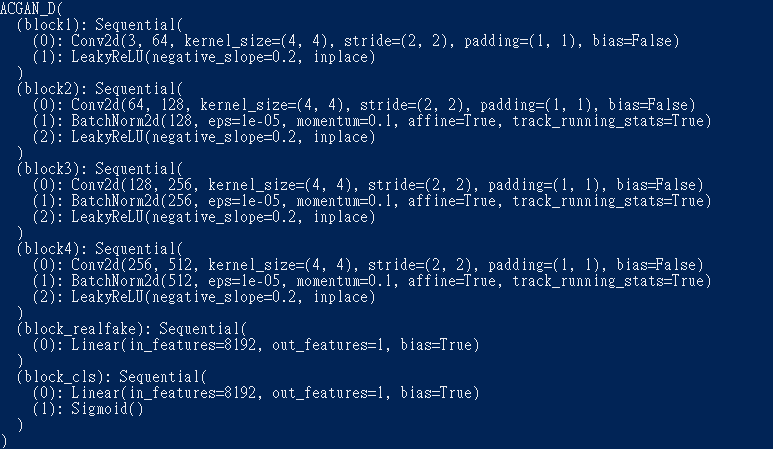
二：WGAN的架構其實論文中並沒有限制。主要分成兩大類，第一類是利用fully connected layer去完成整個架構，第二類是用convolution的做法來完成架構。而其中fully connected架構所做出來的圖片整體看起來會比較smooth，也就是人的臉型以及臉型旁邊的圖像會變得非常不清楚且模糊，但反之train的速度會比較快。第二類的convolution架構會使得整個圖片的輪廓更為清晰清楚，也更有真人的感覺，但train的時間大概會多1.3~1.5倍左右。所以在圖片真偽的考慮上，我最後選擇用convolution的方法去拿到更好的圖片品質。

三：WGAN還有一個進階的做法，叫WGAN-GP (gradient penalty)。他的想法是來自於WGAN實作中有利用Clamp去消除更新過度的數值。而這步驟其實會讓一些發散的數值無法學到新的進步。所以GP就是利用random生成一個真假圖片的interpolation圖片來補償其中資訊的損失，也就是創造更多data去訓練。但這一過程我實作之後發現，所需的train時間會比原本的WGAN多上2倍多，是非常耗時的，要train到好大概要花兩天的時間，所以在時間考量下我捨棄了GP的做法，但GP的圖片品質會比WGAN更有輪廓感，看起來也更真實，真是可惜啊!

Problem 2：ACGAN

1. Describe the architecture & implementation details of your model.





我實作的架構如上圖所示。

ACGAN\_G代表ACGAN的Generator。他是將(batch\_size, 101)的資料轉化成(batch\_size, 3, 64, 64)的一堆假圖片，其中input比GAN多了一維，就是要用來綁定特定的class數值。實作細節是先用fully connected把101維變成512\*4\*4維，再利用convTranspose把512 -> 256 -> 128 -> 64 -> 3 channel。

ACGAN\_D代表ACGAN的Discriminator。他是將(batch\_size, 3, 64, 64)的資料轉化成(batch\_size, 2)的資料型態，其中多出來的1也是用來分辯特定class的不同型態，以笑為例，多出來的一維就是來分辨笑或不笑。而其架構細節就是利用conv把吃進來的3 -> 64 -> 128 -> 256 -> 512 channel，再來分別把 512\*4\*4 做fully connected到1維的資料，再把其中一個1維資料拿去過sigmoid以便分辨class的不同型態。

1. Plot 10 random pairs of generated images from your model.



我選擇的class是笑與不笑這個class。上面是我做出來的圖片，上排是不笑的十個人，下排是這十個人笑起來的樣子。

1. Discuss what you’ve observed and learned from implementing ACGAN.

我認為我學到兩件事。

一：ACGAN本身也沒決定架構該用什麼，所以我將WGAN與ACGAN做結合，在WGAN的loss上面加入BCEloss來衡量class之間的difference，如此一來train出來的loss會變得有意義且可觀察，整個train個過程也會比較穩定。

二：ACGAN的圖片品質因為加入了fully connected layer所以稍微模糊一些，但整體看起來還是有深的輪廓，也有些照片是比較像真人，所以整個實作下來的感覺是還蠻有成就感的。

Problem 3：DANN

1. Compute the accuracy on target domain, while the model is trained on source domain only.

|  |  |
| --- | --- |
| type  data | lower bound |
| USPS ->  MNIST-M | 24.66 |
| MNIST-M  -> SVHN | 33.33 |
| SVHN ->  USPS | 71.99 |

1. Compute the accuracy on target domain, while the model is trained on source and target domain.

|  |  |
| --- | --- |
| type  data | model is trained on source and target |
| USPS ->  MNIST-M | 34.02 |
| MNIST-M  -> SVHN | 47.88 |
| SVHN ->  USPS | 41.60 |

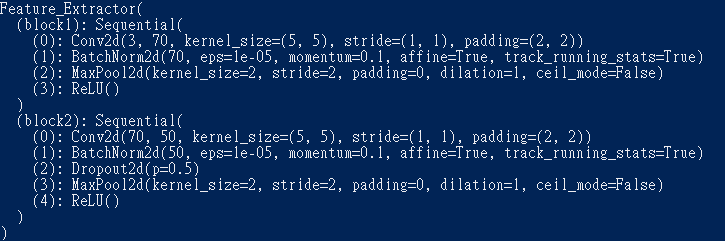
1. Compute the accuracy on target domain, while the model is trained on target domain only.

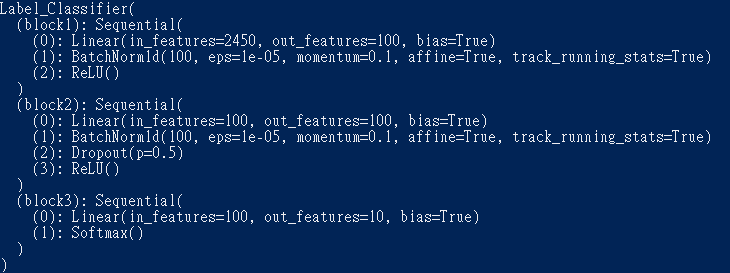
|  |  |
| --- | --- |
| type  data | upper bound |
| USPS ->  MNIST-M | 98.27 |
| MNIST-M  -> SVHN | 93.58 |
| SVHN ->  USPS | 97.65 |

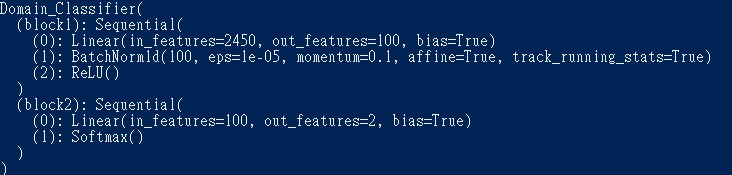
1. Visualize the latent space by mapping the testing images to 2D (with t-SNE) (a)different digit classes 0-9 (b) different domains

|  |  |  |
| --- | --- | --- |
| type  data | different digit classes 0-9 | different domain |
| USPS -> MNIST-M | C:\Users\user\AppData\Local\Microsoft\Windows\INetCache\Content.Word\case1_1-tsne.png |  |
| MNIST-M  -> SVHN |  |  |
| SVHN -> USPS |  |  |

1. Describe the architecture & implementation detail of your model.







以上是我實作的model架構。主要分成三個部分。

Feature\_Extractor：

吃進(batch\_size, 3, 28, 28)輸出成(batch\_size, 50, 7, 7)的Feature，主 要是利用conv把channel 3 -> 70 -> 50。

Label\_Classifier：

吃進(batch\_size, 50\*7\*7)輸出成(batch\_size, 10)，其中10是用來區

分0~9哪個digit。主要是利用fully connected layer把50\*7\*7 ->

100 -> 100 -> 10。

Domain\_Classifier：

吃進(batch\_size, 50\*7\*7)輸出成(batch\_size, 2)，其中2是用來區分

source和target兩個domain。主要是利用fully connected layer把

50\*7\*7 -> 100 -> 2。

1. Discuss what you’ve observed and learn from implementing DANN.

我學習及觀察到的有以下兩點。

一：在做testing的過程中，SVHN -> USPS的lower bound會異常高，而在t-SNE上也可以看出來target的data反而分的比source的data更好，可以推測是整個USPS data跟model的架構向性比較合，可以讓USPS data在數量比較上的情況下依然train得很好，更甚之影響了SVHN data原本應有的分類情況。

二：我學到DANN真的是一個很難的技術，可能最大的原因是每個dataset有他們各自最適合的layer擺法。因為我的layer幾乎是參照網路上別人train得很好的model去做訓練，但套到我們這三組dataset上之後就幾乎沒有這麼好的效果，更甚至t-SNE的圖看起來並有效。所以推測可能最好的DANN作法是依據不同model特別去試各種不同的layer擺法，才能各自有最好的效果。

Problem 4：DAN

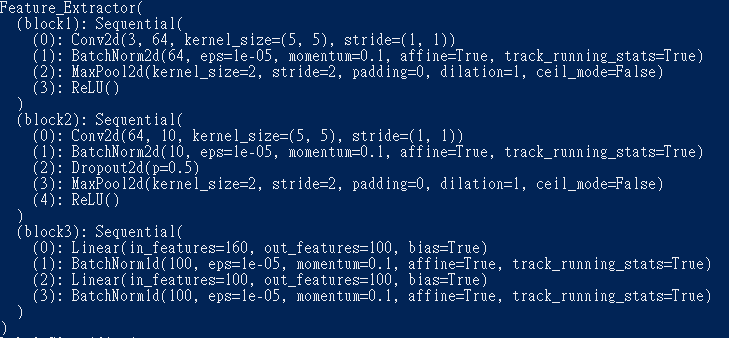
1. Compute the accuracy in target domain, while the model is trained on source and target domain.

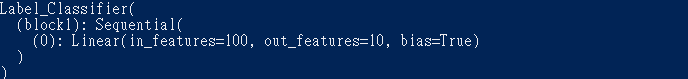
|  |  |
| --- | --- |
| type  data | model is trained on source and target |
| USPS ->  MNIST-M | 37.26 |
| MNIST-M  -> SVHN | 48.32 |
| SVHN ->  USPS | 62.83 |

1. Visualize the latent space by mapping the testing images to 2D (with t-SNE) (a)different digit classes 0-9 (b) different domains

|  |  |  |
| --- | --- | --- |
| type  data | different digit classes 0-9 | different domain |
| USPS -> MNIST-M |  |  |
| MNIST-M -> SVHN |  |  |
| SVHN -> USPS |  |  |

1. Describe the architecture & implementation detail of your model.





我實作的是DAN，以上是我實作的model，主要分成兩個部分。

Feature\_Extractor：

吃進(batch\_size, 3, 28, 28)輸出成(batch\_size, 100)的Feature。主要

是利用conv把channel 3 -> 64 -> 10，再利用fully connected layer 把10\*4\*4 -> 100 -> 100。

Label\_Classifier：

吃進(batch\_size, 100)輸出成(batch\_size, 10)。其中10是用來區分 0~9的哪個digit。架構上是利用fully connected layer把100 -> 10 維。

1. Discuss what you’ve observed and learn from implementing your improved UDA model.

我所實作的improved是DAN，我認為我學到及觀察到的東西有以下兩點。

一：

DAN架構跟DANN的差別在於，DAN沒有一個domain classifier，而有兩個source跟target的label classifier，以及一個feature extractor。

他是在原本source label loss的基礎上加上mmd loss，用以衡量source\_features和target\_features以及source\_pre\_labels和target\_pre\_labels之間的距離，想辦法讓兩個距離更靠近，再加上原本就有的source label loss，可以讓source的label\_classifier更精準地拉近source與target。

二：

DAN的t-SNE以及accuracy都要比原本的DANN還要好上許多。從t-SNE圖片可以看得出來，兩個domain彼此有在靠近，而且同類別的class是有群聚效果的，所以可以顯示說DA在這部份是非常成功的，整題看起來地靠近程度也比DANN好上許多。

合作同學：

我整份報告有疑問的地方，都是與我 ImLab 的實驗室同學們一起討論，我們僅限於題目定義與演算法的討論，實作的 code 都是各自寫各自的。以下是同學們的 ID，b04901190、R07922002、R07922024、R07922043、R97922120。

參考資料：

WGAN：

[1] <https://zhuanlan.zhihu.com/p/25071913>

[2] <https://github.com/eriklindernoren/PyTorch-GAN>

[3] <https://www.jianshu.com/p/ddfd7fba11d0>

ACGAN：

[1]<https://zhuanlan.zhihu.com/p/44177576>

DANN：

[1] <https://github.com/NaJaeMin92/pytorch_DANN/blob/master/model.py>

[2] <https://github.com/CuthbertCai/pytorch_DANN>

DAN：

[1] <https://github.com/CuthbertCai/pytorch_DAN/blob/master/models.py>

[2] <https://discuss.pytorch.org/t/maximum-mean-discrepancy-mmd-and-radial-basis-function-rbf/1875?fbclid=IwAR0l6956_05qOXIhBJg9eOltTWh870P5lFgOCOHZl_QwHZ8wo3ppE2JSX1E>