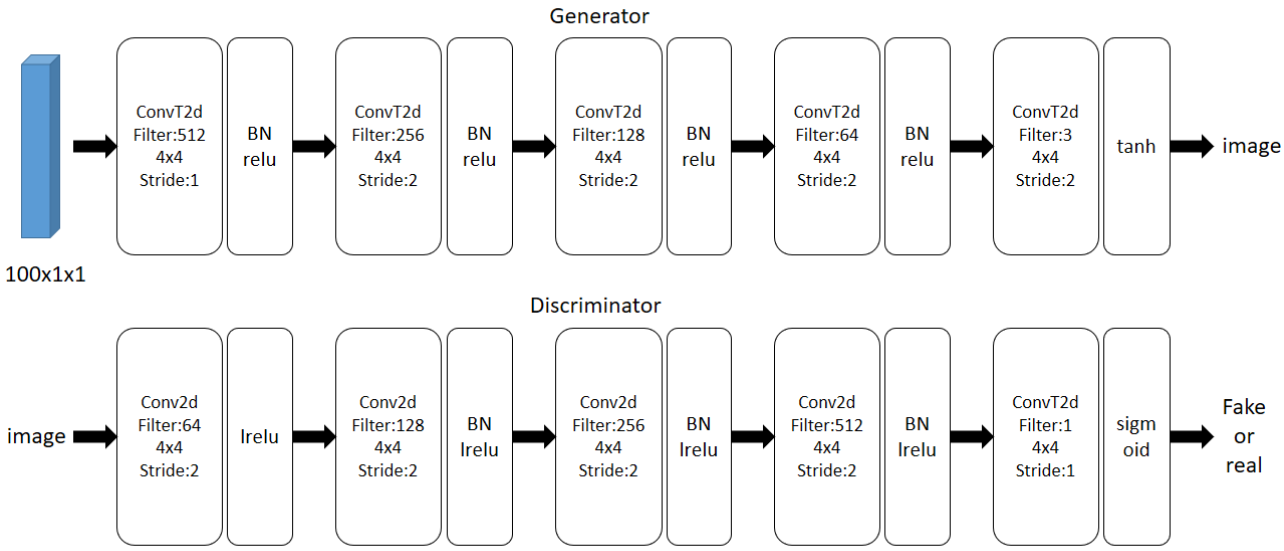
DLCV HW3

鄭承昀 機械碩一 R07522823

Problem 1 GAN

1. Describe the architecture and implementation details of ypur model.



這邊我採用的是DCGAN的架構。

Generator的部份，輸入一個100x1x1的Gaussian randoom noise，經過transpose convolution layers、batch normalization和relu。最後經過Tanh使輸出介於-1到1之間，在經過轉換使其介於0到1之間。

Discriminator的部份則是類似一般的CNN，輸出為兩類的classification。

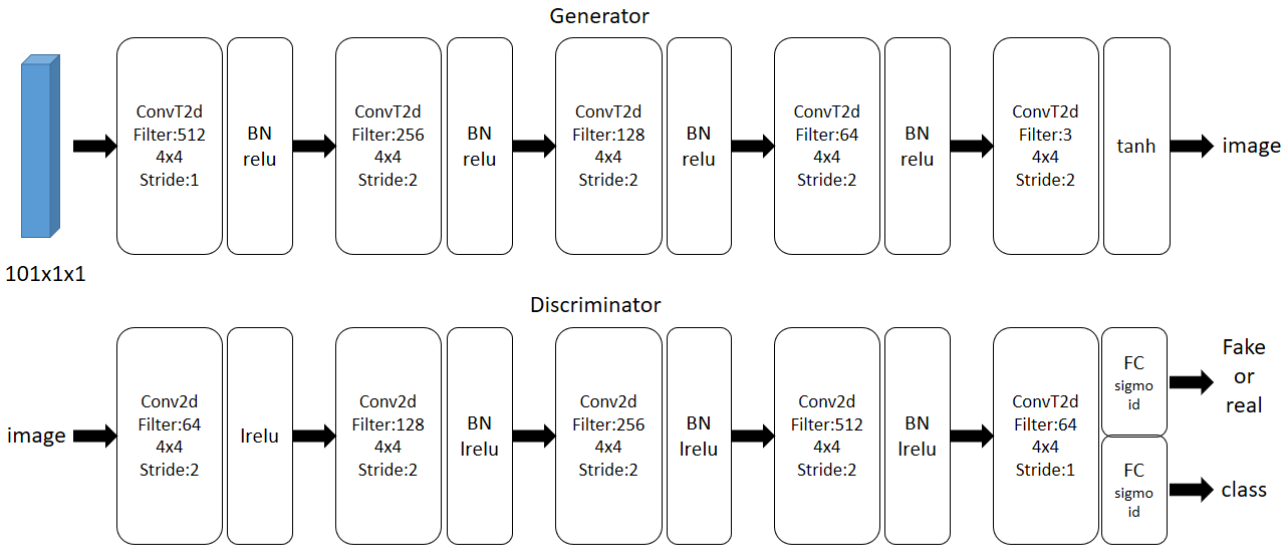
Optimizer為Adam，learning rate = 0.0002、beta1 = 0.5、beta2 = 0.999，loss function採用binary cross-entropy loss。

2. Plot 32 random images generated from your model.

3. Dicuss what you've observed and learned from implementing.
    - 參考了網路上的一些架構設計、參數設定讓我很快就能生成出一些看起來還行的圖片。
    - 我發現在有些人像在很初期的時候就已經生成得很好了，但是相對的有些初期生成特別爛的人像在經過完整的training後還是無法生成的很好。
    - 調整discriminator和generator 更新的次數能使圖片生成的更好。一開始我兩者更新相同的次數，我發現到後面generator的loss變得比較高。所以我改成discriminator更新一次，generator更新兩次，generator的loss因此下降，圖片生成的結果也比較好。
4. References
    - https://github.com/thtang/DLCV2018SPRING/tree/master/hw4
    - https://github.com/pytorch/examples/tree/master/dcgan

Problem 2 ACGAN

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



這裡架構基本上跟上面所使用的DCGAN一模一樣，其中有差別的部份在於：

- Generator的輸入改成101x1x1，其中加入的那一維即為我們所期望的attribute。
- Discriminator最後經過兩個不同的fully-connected layer，分別去辨別Fake or real、smile or not。

Optimizer為Adam，learning rate = 0.0002、beta1 = 0.5、beta2 = 0.999，loss function採用binary cross-entropy loss。

2. Plot 10 random pairs of generated images from your model, where each pair should be generated from the same random vector input but with opposite attribute. This is to demonstrate your model's ability to disentangle features of interest.

   Smiling or not

3. Discuss what you've observed and learned from implementing ACGAN.

   - 因為基本上架構和參數都是依照上面一題的DCGAN去做調整，在生成圖片上比較沒有太大的問題。
   - 儘管有些人像的生成並不是非常的好，但是在attribute的表現上基本都可以看出差異。我猜測原因為attribute算是supervised的，而人臉的生成則是unsupervised的。而supervised的效果高於unsupervised。

4. Reference

   - https://github.com/thtang/DLCV2018SPRING/tree/master/hw4

Problem 3 DANN

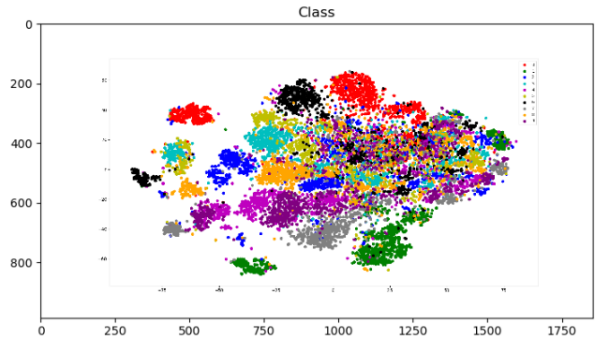
1. Compute the accuracy.

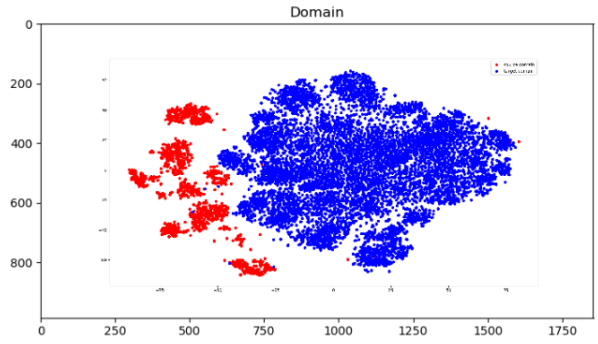
	USPS -> MNIST-M	MNIST-M -> SVHN	SVHN -> USPS
Trained on source	23.36%	24.29%	67.61%
Adaptation(DANN)	44.42%	51.36%	57.70%
Trained on target	98.45%	92.77%	97.26%

2. Visualize the latent space by mapping the testing images to 2D space (with t-SNE) and use different colors to indicate data of (a) different digit classes 0-9 and (b) different domains.

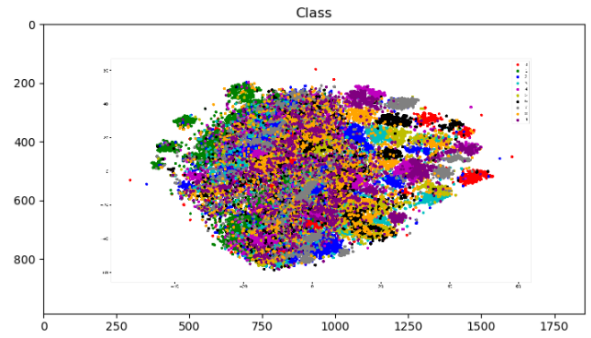
   這邊我選用最後一層Conv的output作為feature。

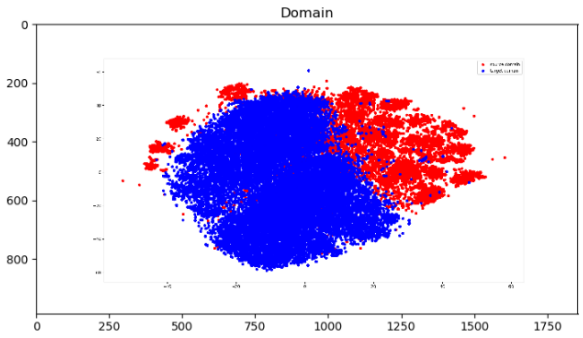
   - USPS -> MNIST-M



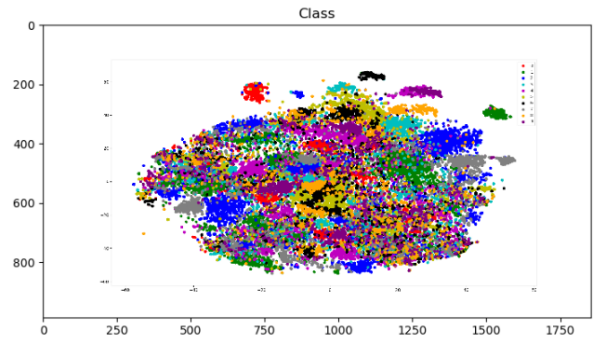


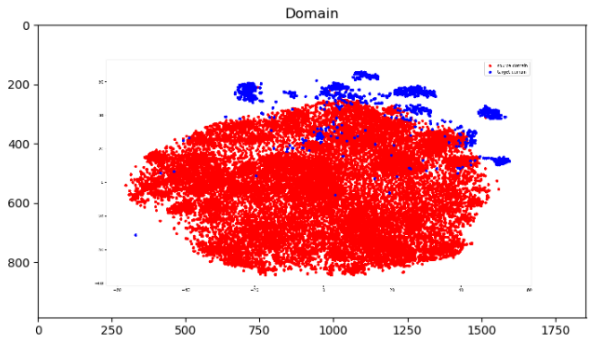
   - MNIST-M -> SVHN



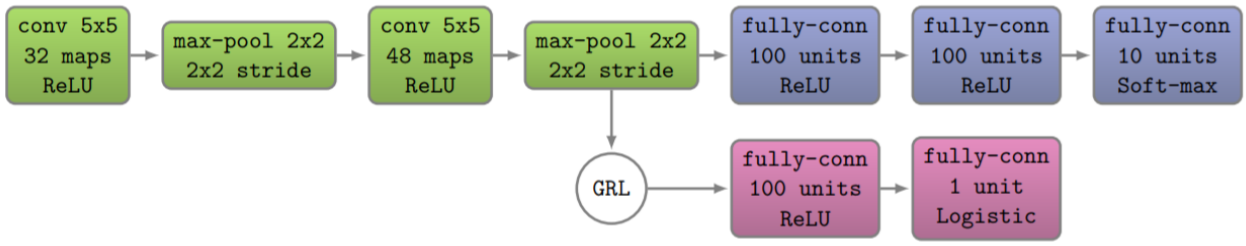


   - SVHN -> USPS





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



這張架構圖是從DANN的paper上截下來的，而我就是採用這個架構去做實現。其中，feature extractor為基本的Conv和max-pool，label predictor為三層fully-connected layer，domain predictor則為兩層的fully-connected layer。

Optimizer為Adam，learning rate = 0.001，domain和label的loss function皆採用cross-entropy loss。

4. Discuss what you've observed and learned from implementing DANN.

   - 在做DANN其實沒有遇到太多的問題，依照paper上面的架構來做實現即成功超過baseline。
   - 從上面t-SNE的圖上發現分類的結果其實蠻不好的。Class的部份並沒有成功分開，domain的部份看起來也是各為一區。
   - 其中有一個值得注意的點，當只用SVHN來做training時，在USPS的正確率比同時使用SVHN+USPS的DANN來的更高。其可能原因為，我在train DANN時使用的data量為data較少的那個（即min(len(source_trainloader), len(target_trainloader))），而SVHN和USPS的data量相差甚大（差大約10倍）。因此，只用SVHN做training時，因為data量較大，model可能學到較好的feature，使得在其他 domain上也有一定程度的正確率。

5. Reference

   - https://github.com/jindongwang/transferlearning/tree/master/code/deep/DANN(RevGrad)
   - https://github.com/fungtion/DANN

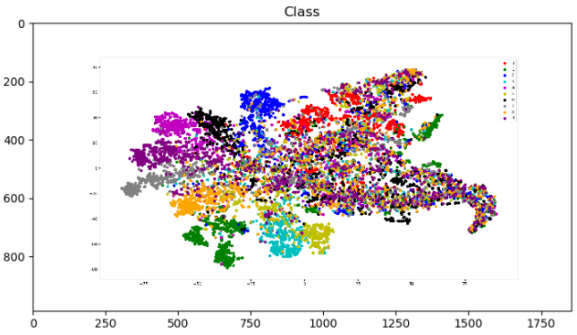
Problem 4 Improved UDA model

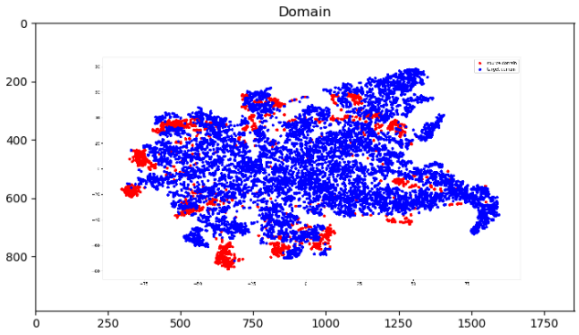
1. Compute the accuracy on target domain, while the model is trained on source and target domain.

	USPS -> MNIST-M	**MNIST-M -> SVHN**	**SVHN -> USPS**
Adaptation(ADDA)	34.42%	45.37%	38.96%

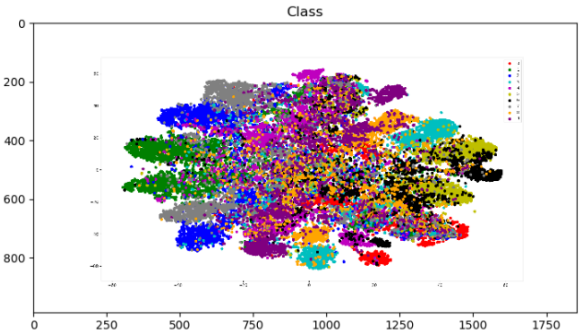
2. Visualize the the latent space by mapping the testing images to 2D space (with t-SNE) and use different colors to indicate data of (a) different digits classes 0-9 and (b) different domains (source/target).

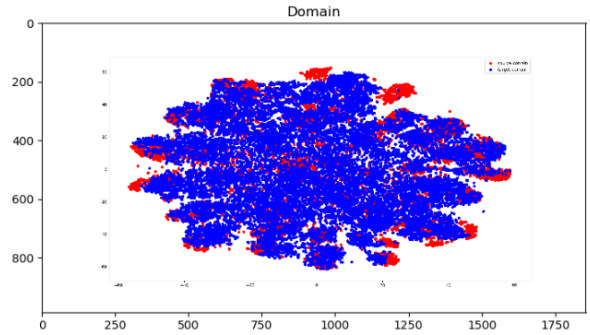
   - USPS -> MNIST-M



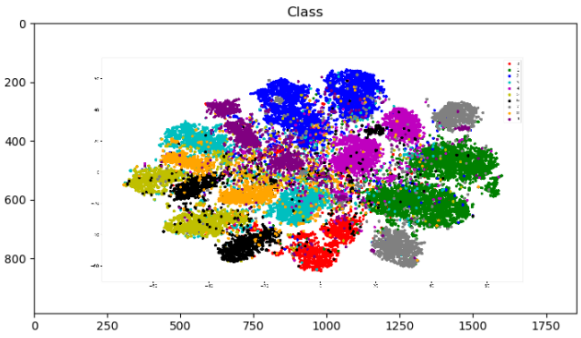


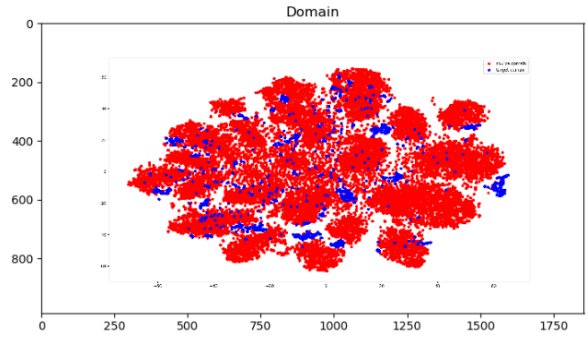
- MNIST-M -> SVHN



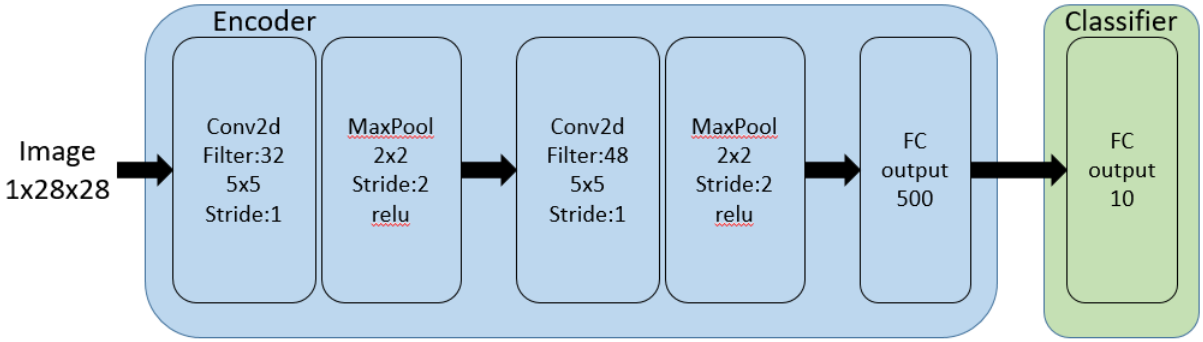


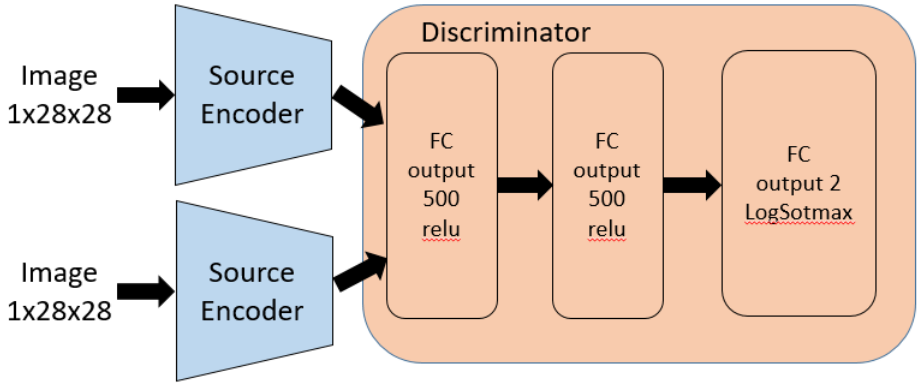
- SVHN -> USPS





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





這題我採用的是ADDA。 其中，source、target的encoder僅為簡單的兩層Conv+MaxPool以及一個fully-connected layer。 Classfiler僅為單層的fully-connected layer。 Discriminator則為三層fully-connected layer。

不論是pretrain source model還是train target encoder，optimizer都是使用的是Adam，learning rate = 0.0001、beta1 = 0.5、beta2 = 0.9，loss function採用binary cross-entropy loss。

4. Discuss what you've observed and learned from implementing your improved UDA model.

   - 其實從上面的正確率來看，ADDA的結果都比DANN來的差。但是從t-SNE圖的結果上我卻覺得分類的結果好像更好。同樣class的分佈比較密集，不像DANN那樣什麼都看不出來。
   - 在做target encoder和discriminator的training時，我發現我在testset上的正確率會有稍微下降的趨勢。我猜測是因為discriminator會同時使用到source data和target data，其訓練量為target encoder的兩倍。所以我將target encoder每次更新的次數設為2，讓其訓練量相同。
   - 除了ADDA，我還嘗試用DSN，但是我train出來的結果都只有10%到20%得正確率。

5. Reference

   - https://github.com/corenel/pytorch-adda