名字： 劉兆鵬 系所：電機所碩一 學號：R07921052

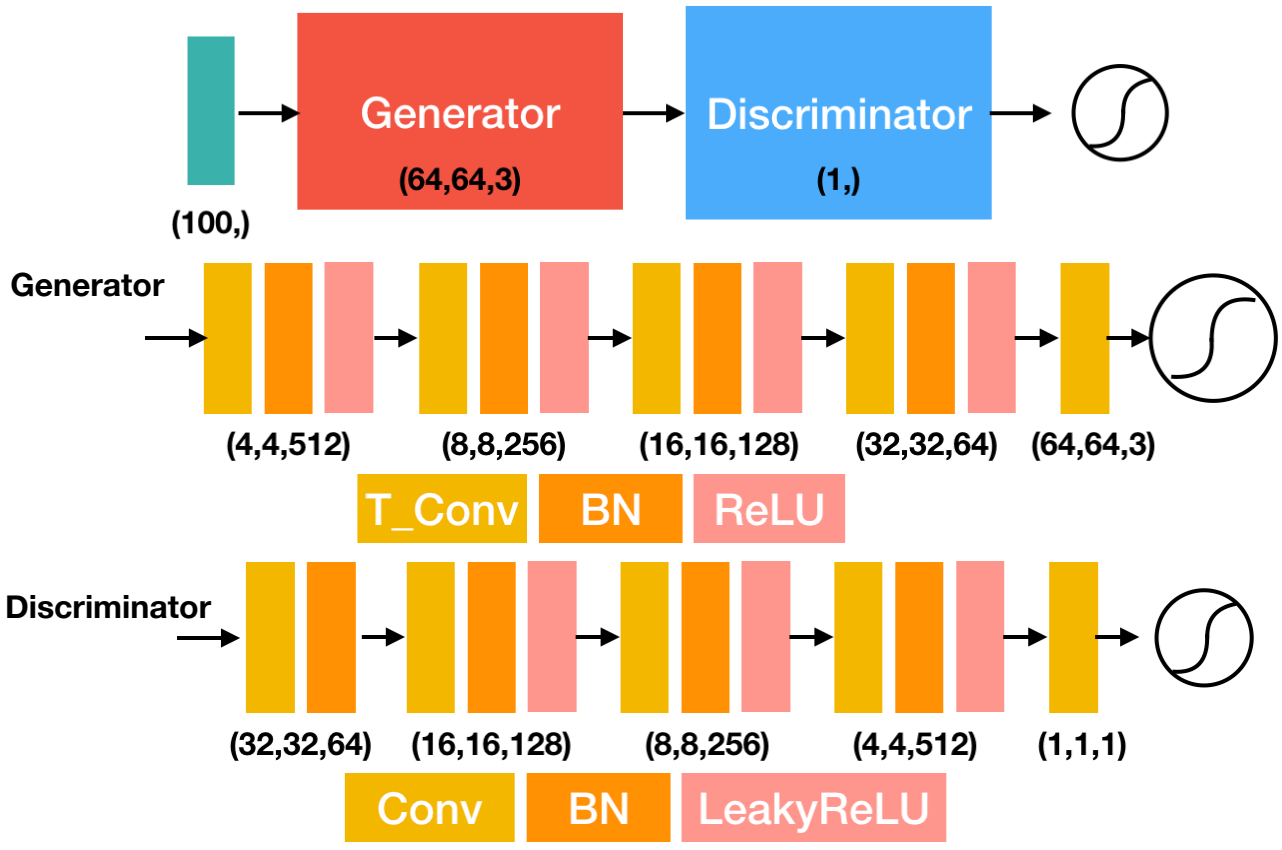
Problem 1: GAN

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

在實作 GAN 時我所使用的架構分為 Generator 以及 一個 Discriminator 進行圖片生成及辨識其真偽。其架構如下圖一所示。其為使用一個隨機產生的 Gaussian noise 輸入至 Generator 中，而我的 Generator 為使用多組由 transposed convolution, batch normalization 以及 relu activation function 的 block 所組成如下圖二，其為將 noise 由較小的向量通過多組 transpose convolution 進行放大後，形成與目標圖像相同大小的圖片，而在最後一項 transposed convolution 所串接的 activation function 為 tanh，使得輸出之圖像範圍介於 -1 到 1 之間。然而，經過我多次的實驗後，我發現讓 Generator 產生出的圖片應轉換為 0 至 1 之間，會使得圖片的品質較佳，故在 generator 產生出的圖片我將其乘以 2 並加上 0.5 使其值的範圍轉為 0~1 之間，再輸入至 Discriminator 進行辨識圖片的真偽。

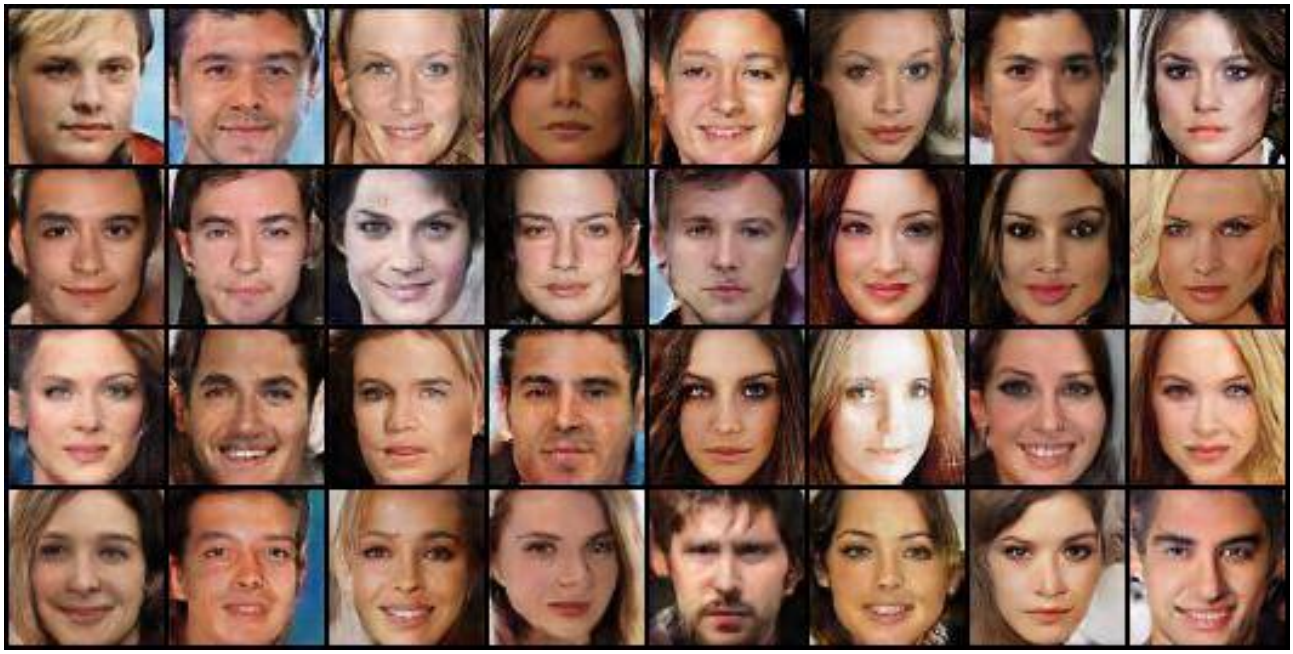
將 Generator 所生成的圖像輸入至 Discriminator 後，我一樣使用多組 Convolution 以及 LeakyReLU 的 activation function 所組成的 block 進行圖像 downsample 的處理如下圖三所示，而在此所使用的參數皆與 Generator 雷同，目的是為了讓 Generator 與 Discriminator 在每一層 block 所增加與減少的維度為相同的，經過實驗後發現讓兩者在每一層所輸出的維度相同時，能夠取得不錯的圖像品質。

接著我將原始的圖片也輸入至 Discriminator 中，使 Discriminator 了解真實圖片的樣貌，便能進行真實圖片及生成圖片的辨識訓練。而在調整 Discriminator 一次後，我會去調整 Generator 的參數，此時為將 Generator 所輸出的圖片的 label 標為真實圖片，使 Generator 能夠逐漸變好至能夠輸出與真實圖片相似的影像。



2. Plot 32 random images generated from your model. [fig1_2.jpg]

根據上述所訓練完的結果，我只使用 Generator model 的部分，隨機產生一些 noise 後輸入至 Generator 中，便能產生出與目標相似的圖像資料，其結果如下所示。



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

在這次訓練 GAN 時，我發現此 task 對於參數的調整以及模型的架構都十分的敏感，對於 learning rate 或是 activation function 以及輸入的 input 值範圍都會有很大的影響，像是我一開始將 generator 的值介於 -1~1 之間就會讓圖像的品質較差，改為 0~1 後就能夠明顯提升 generator 的輸出品質。而對於 input 要介於 -1~1 或是 0~1 也是有許多差別，故訓練 GAN 讓我覺得每一個參數都需要慢慢的去調整，找到最佳的結果。且在訓練的過程也發現 learning rate 需要隨著訓練的過程跟著一起去調整，不然圖像的品質不會持續進步，也讓我實驗嘗試了一段時間。最後在 generator 以及 discriminator 訓練次數比例的調整也讓我嘗試了很多次，一開始讓兩者在每個 epoch 訓練的次數相同時，都會導致圖片 generator 輸出的品質不佳，我以為是因為 discriminator 不夠強，所以使得 generator 在品質不佳的情況下就能夠騙過 discriminator，故增加了 discirminator 的訓練次數，但效果依然有限，後來發現是要將generator訓練次數多增加一點，發現能夠讓輸出的品質變好。我想這些都是要根據不同的 task 來進行調整的參數，故在 GAN 生成圖像上是需要慢慢調整參數的。

Problem 2: ACGAN

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

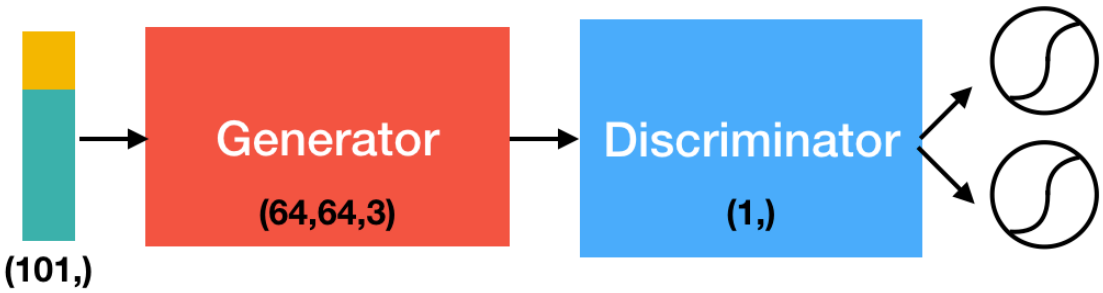
在實作 ACGAN 時我所使用的架構也是分為 Generator 以及 一個 Discriminator 進行圖片生成及辨識其真偽。其架構如下圖一所示。其為使用一個隨機產生的 Gaussian noise 並加上我隨機產生的 class 資訊，在此我使用 smiling or not smiling 的特徵作為的 auxiliary classifier 所訓練的特徵。

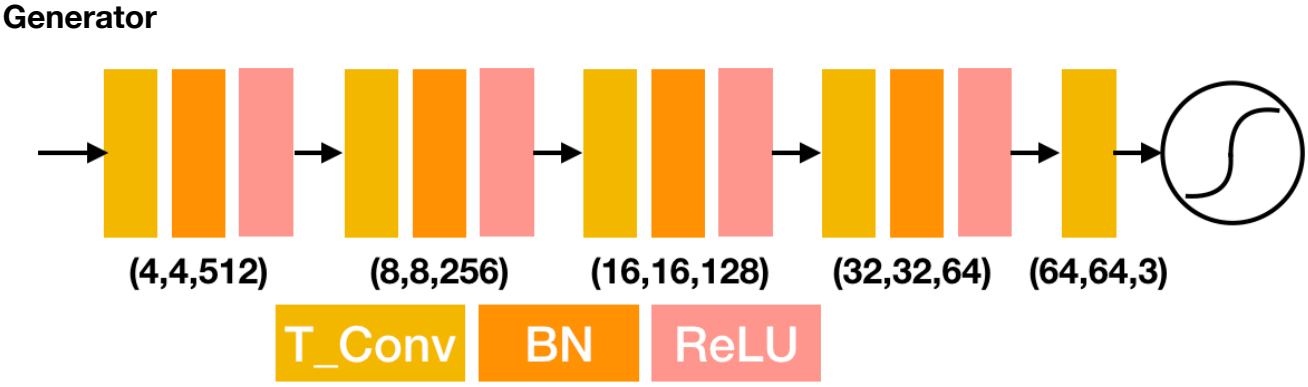
將隨機產生的 noise 以及 class 接合後輸入至 Generator 中，而我的 Generator 為使用多組由 transposed convolution, batch normalization 以及

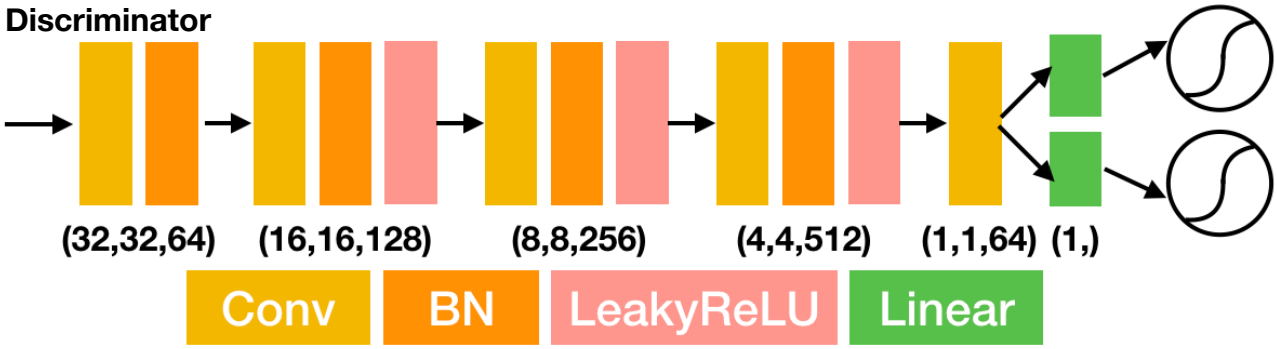
relu activation function 的 block 所組成如下圖二，其為將 noise 由較小的向量通過多組 transpose convolution 進行放大後，形成與目標圖像相同大小的圖片，而在最後一項 transposed convolution 所串接的 activation function 為 tanh，使得輸出之圖像範圍介於 -1 到 1 之間。然而，經過我多次的實驗後，我發現讓 Generator 產生出的圖片應轉換為 0 至 1 之間，會使得圖片的品質較佳，故在 generator 產生出的圖片我將其乘以 2 並加上 0.5 使其值的範圍轉為 0~1 之間，再輸入 Discriminator 進行辨識圖片的真偽以及辨識圖片的類別。

將 Generator 所生成的圖像輸入至 Discriminator 後，我一樣使用多組 Convolution 以及 LeakyReLU 的 activation function 所組成的 block 進行圖像 downsample 的處理如下圖三所示，而在此所使用的參數皆與 Generator 雷同，目的是為了讓 Generator 與 Discriminator 在每一層 block 所增加與減少的維度為相同的，經過實驗後發現讓兩者在每一層所輸出的維度相同時，能夠取得不錯的圖像品質。接著在最後一層輸出的特徵串接上兩個不同 Linear layer，將特徵分別進行真偽的辨識以及辨識此生成的圖片是否與預期的類別相同。

接著我將原始的圖片也輸入至 Discriminator 中，使 Discriminator 了解真實圖片的樣貌，以及了解圖片對應的類別為何，便能進行真實圖片及生成圖片的真偽辨識及類別訓練。而在調整 Discriminator 一次後，我會去調整 Generator 的參數，此時為將 Generator 所輸出的圖片的 label 標為真實圖片以及將其類別對應預設的類別，使 Generator 能夠逐漸變好至能夠輸出與真實圖片相似的影像且符合我預設的類別。







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 if to demonstrate your model's ability to disentangle features of interest. [fig2_2.jpg]

根據上述所訓練完的結果，我只使用 Generator model 的部分，隨機產生十組 noise 並使用不同的 class attribute 後輸入至 Generator 中，便能產生出十組相同的 pair 但其對應的 class 為不同的圖像。其結果如下圖所示。

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

在這次訓練 ACGAN 時，我發現此 task 對於參數的調整以及模型的架構都十分的敏感，對於 learning rate 或是 activation function 以及輸入的 input 值範圍都會有很大的影響，像是我一開始將 generator 的值介於 -1~1 之間就會讓圖像的品質較差，改為 0~1 後就能夠明顯提升 generator 的輸出品質。而對於 input 要介於 -1~1 或是 0~1 也是有許多差別，故訓練 ACGAN 讓我覺得每一個參數都需要慢慢的去調整，找到最佳的結果。且在訓練的過程也發現 learning rate 需要隨著訓練的過程跟著一起去調整，不然圖像的品質不會持續進步，也讓我實驗嘗試了一段時間。最後在 generator 以及 discriminator 訓練次數比例的調整也讓我嘗試了很多次，一開始讓兩者在每個 epoch 訓練的次數相同時，都會導致圖片 generator 輸出的品質不佳，我以為是因為 discriminator 不夠強，所以使得 generator 在品質不佳的情況下就能夠騙過 discriminator，故增加了 discirminator 的訓練次數，但效果依然有限，後來發現是要將generator訓練次數多增加一點，發現能夠讓輸出的品質變好。我想這些都是要根據不同的 task 來進行調整的參數，故在 ACGAN 生成圖像上是需要慢慢調整參數的。另外我也發現 ACGAN 所輸出的圖像品質相較於 GAN 本身是有提升的，我認為透過更多的 class 資訊能夠讓 Discirminater 以及 Generator 擁有更多的資訊來生成我們我目標的圖像，故我觀察到在訓練生成圖片時，若能給予更多資訊就應給予使其品質提升。

Problem 3: DANN

1. Compute the accuracy on target domain, while the model I trained on source domain only. (Lower bound)

在此題我為將 model 分別訓練於三個 task 的 dataset 上，並在訓練完後將其各自 model 預測於其對應的 target dataset 上，其結果如下所示。

	USPS->MNISTM	MNISTM->SVHN	SVHN->USPS
Train on source	0.3389	0.36678	0.6721

2. Compute the accuracy on target domain, while the model is trained one source and target domain. (domain adaption)

在此題我為將 model 分別訓練於三個 task 的 dataset 上，並使用 DANN 來將 source 與 target data distribution 拉近，並於訓練完後預測於對應的 target dataset 上，分別對應了 usps->mnistm, mnistm->svhn, svhn->usps，其結果如下所示。

	USPS->MNISTM	MNISTM->SVHN	SVHN->USPS
Adaption(DANN)	0.4984	0.54433	0.6751

3. Compute the accuracy on target domain, while the model is trained on source and target domain only. (Upper bound)

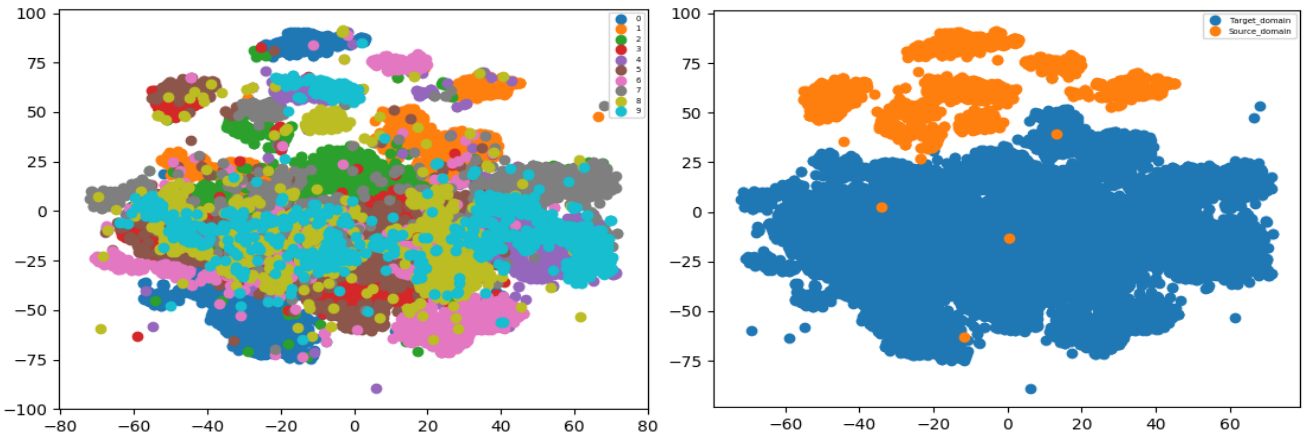
在此題我為將 model 分別訓練於三個 task 的 target dataset 上，並使用 target dataset 作為訓練的 training data，並於訓練完後預測於自身的 target dataset 的 test set 上，其結果如下所示。

	USPS->MNISTM	MNISTM->SVHN	SVHN->USPS
Trained on target	0.9841	0.9304	0.9740

4. 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 domain (source/target).

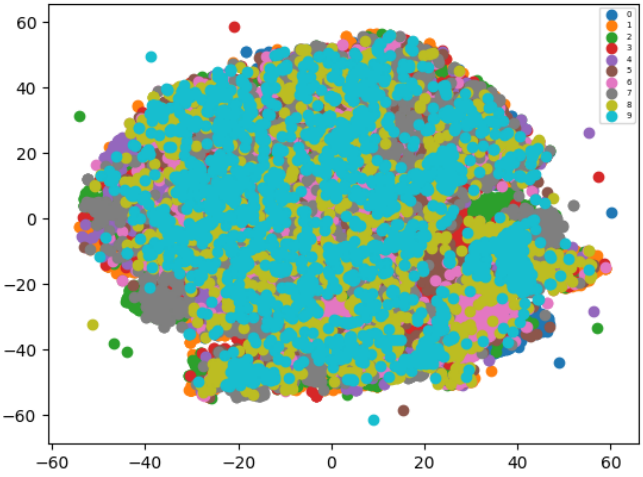
在此其題我將我所訓練完的 DANN model 擷取此前半段的 encoder 部分，將最後一層輸出作為我的 latent space，我將各自 adaption model 將 source 與 target 的 testing set 的 feature 輸出，並透過 t-SNE 降維畫出兩者的分佈，分別將其各自的 digit classes 以及 domain 畫出，來觀察 DANN 是否能夠拉近不同 dataset 的距離。其結果如下所示。

USPS->MNISTM

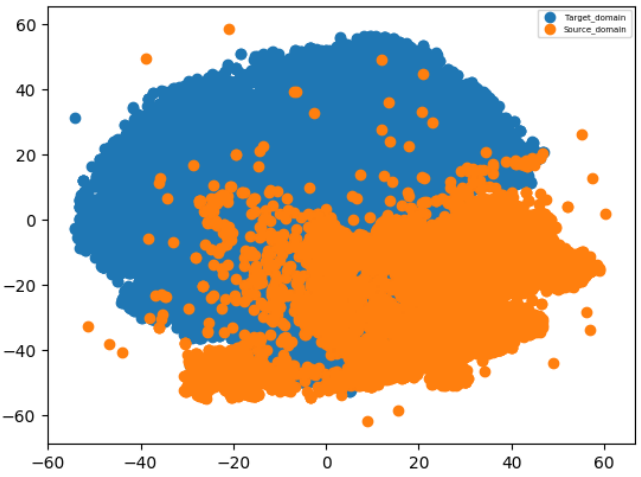


(a) (b)

MNISTM->SVHN

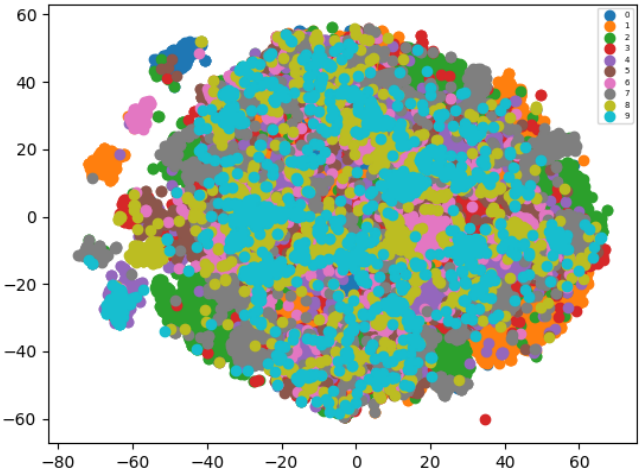


(a)

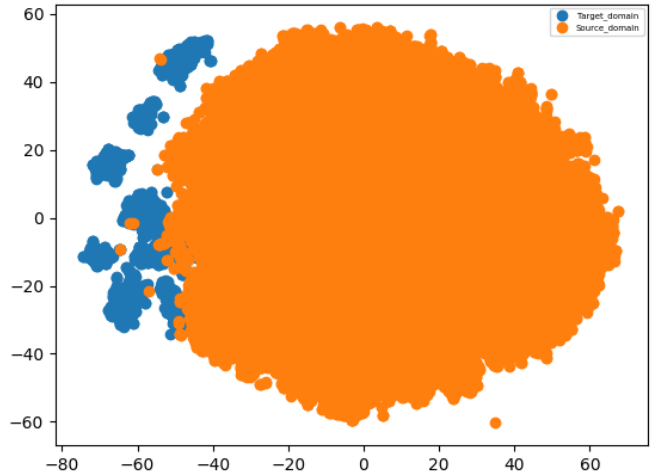


(b)

SVHN->USPS



(a)



(b)

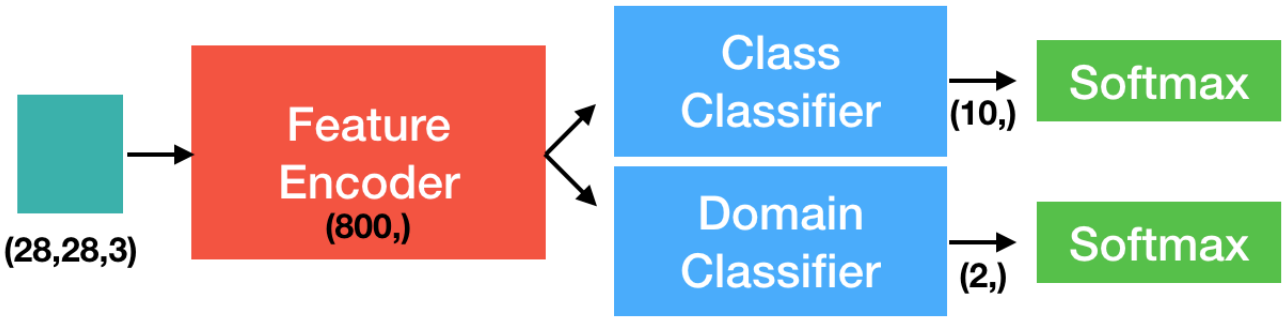
5. Describe the architecture and implementation detail of your model.

DANN 的模型架構可分為 source 和 target data 的 feature encoder，source data 的 class classifier 以及 source and target domain 的 classifier，其整體架構如下所示。

在 feature encoder model 訓練的部分採用了 reversal layer 的方式，也就是將原先需要 update feature encoder layer 的 gradient 取負值的部分，目的是在 feature encoder 的訓練部分要去混淆 source 以及 target data 的分佈，期望透過此方式能夠讓 feature encoder 對 source 以及 target data distribution 能夠合在一起，又會使用 source data 的 class 訓練 class classifier 的部分，故使得在預測時能夠使用 target data 的 feature encoder 配上 source 的 class classifier 來進行預測。

而訓練的過程則是在每一個 epoch 時先使用 source data 計算整體 model 的 gradient，接著在使用 target data 的 image 進行 feature encoder 以及 domain classifier 的 gradient 計算，最後在將總 loss 加總一起做 model 的 update。

另外，在 reversal layer 的部分，gradient 的部份會根據整體 epoch 及 iteration 的進行從 0~1 的增加權重，使 reversal layer 能夠緩慢的訓練，避免初始差異過大。



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

在這次訓練完 DANN 後，並將此結果 feature encoder 的 latent space 的結果畫出後，發現其實兩 dataset 的 distribution 雖然很靠近，然而其在 class 的部分卻無法將各類別清楚地分開，經過實驗以及查詢了一些文獻後，說明此 DANN 的方法因為是將 class classifier 一同訓練，可能因為使用在 digit task 上使得 classifier 的部分很快就收斂了，導致沒有 class 的 gradient 回傳至 feature encoder，所以會使得 class 的分佈無法很清楚的將各類別區分，然而 domain 的部會有持續的訓練，故能夠發現其 domain distribution 是能夠相互靠近的，並使準確率有較 lower bound 來的高。

Problem 4: Improved UDA model

1. Compute the accuracy on target domain, while the model is trained one source and target domain. (domain adaption)

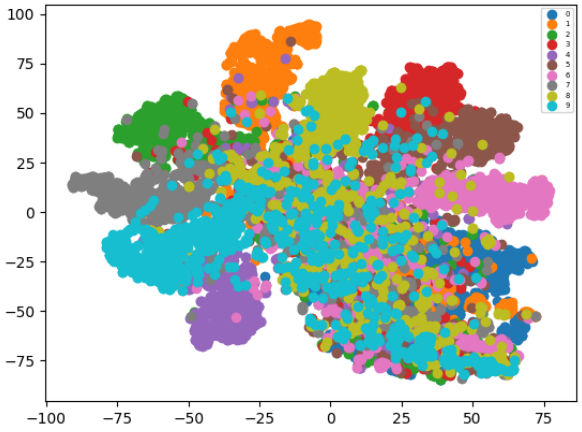
在此題我為將 model 分別訓練於 tasks 的 dataset 上，並使用 ADDA 做為 improved model 來將 source 與 target data distribution 拉近，並於訓練完後預測於對應的 target dataset 上，分別對應了 usps->mnistm, mnistm->svhn, svhn->usps，其結果如下所示。

	USPS->MNISTM	MNISTM->SVHN	SVHN->USPS
Adaption(ADDA)	0.5508	0.5922	0.714499

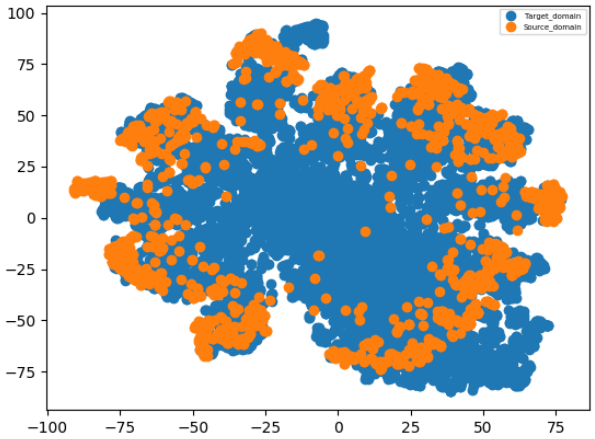
2. Visualize the latent space by mapping the testing images to 2D space(with t-SNE) and use different colors to indicate data of (a) different digit classes 0-9 and (b) different domain (source/target).

在此其題我將我所訓練完的 ADDA model 擷取此前半段的 encoder 部分，將最後一層輸出作為我的 latent space，我將各自 adaption model 將 source 與 target 的 testing set 的 feature 輸出，並透過 t-SNE 降維畫出兩者的分佈，分別將其各自的 digit classes 以及 domain 畫出，來觀察 ADDA 是否能夠拉近不同 dataset 的距離。其結果如下所示。

USPS->MNISTM

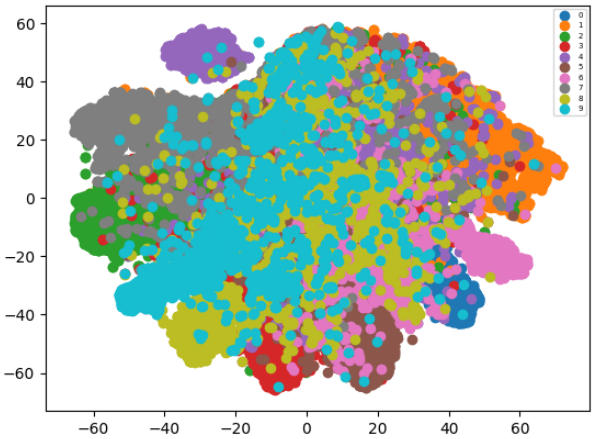


(a)

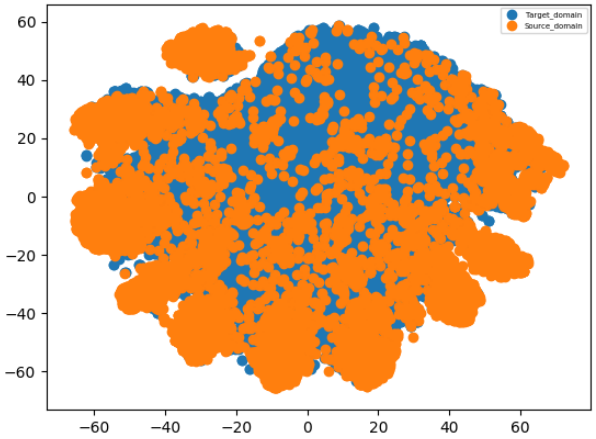


(b)

MNISTM->SVHN

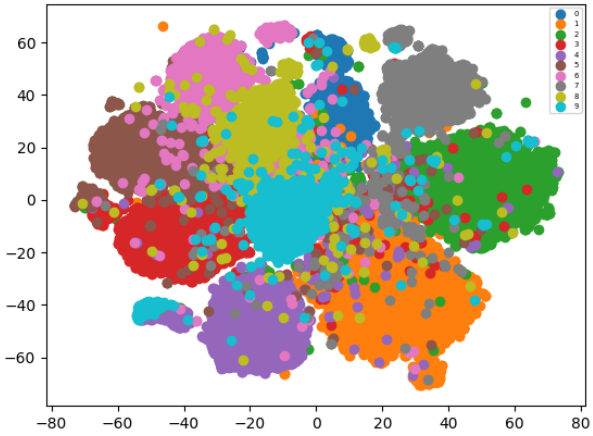


(a)

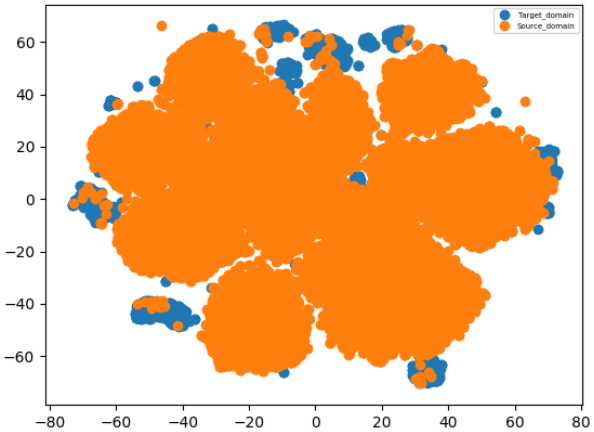


(b)

SVHN->USPS



(a)



(b)

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

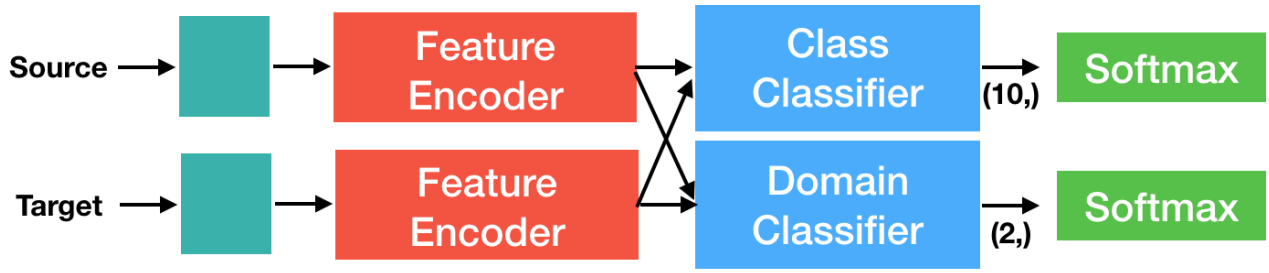
在此題我為實作 ADDA 的架構，其有別於 DANN 將 source 及 target 使用同一個 feature encoder 的方式，ADDA 為使用兩個權重不分享的 feature encoder 來學習 dataset 的分佈。故整體的架構可分為 source data 及 target data 的 feature encoder，也包含 source data 的 class classifier 及 domain classifier，其架構如下所示。

在訓練的過程中，我先訓練 source data 的 feature encoder 以及 class classifier 的部分，主要是先去讓 model 了解說 source data 的分佈及類別的訓練結果，在選取於 source data 最好的 model 後，我就固定住了 source data 的 feature encoder 以及 class classifier 的部分。

接著我將 source data 的 feature encoder 的權重作為 target data 的 feature encoder 的初始值進行訓練，其訓練方式與 GAN 的訓練方法相同，我將 source 的 feature encoder 所輸出的 latent space 作為 real data，而 target feature encoder 所輸出的結果做為 fake data，將兩者皆經由 domain classifier 作為 discriminator，進行 source 及 target 的真偽訓練，使 domain classifier 能夠分辨其為 source 還是 target data。

而接著再訓練 target data 的 feature encoder，期望此 encoder 如同 GAN 的 generator 部分能夠產生出的 latent space 能夠欺騙過 discriminator，使得 target feature encoder 能夠輸出與 source data 有相近的 distribution。

最後訓練完後，將 target feature encoder 接上先前訓練好的 source data class classifier 的部分，進行預測，其效果能夠高於 DANN 的結果且也能將 source 及 target 的分佈拉得更近。



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

在訓練完 ADDA 後，將結果的 latent space 畫出後可以發現其效果較 DANN 要來得明顯，能夠將兩者不同 dataset 的分佈幾乎重疊在一起，且對應的每一個 class 類別也都能夠較明顯的分散，發現此方法主要的目的是有別於 DANN 的作法，其希望的主旨為要將 source 及 target 的 distribution 拉近，希望能夠使用 source data 所訓練好的 classifier 就能夠對 target data 進行預測，故事先將 source data 的 feature encoder 及 classifier 都先訓練後固定其參數，接著訓練 discriminator 以及 target feature encoder，目標要讓其 discriminator 無法分別 source 及 target data，就能夠使得兩者距離靠近了。故能夠使得準確率較 DANN 來得高，使達到 improved 的效果。

7. 參考資料

Github: https://github.com/fungtion/DANN, https://github.com/corenel/pytorch-adda.