學號：r07921001　　系級：電機所二　　姓名：李尚倫

1. (3%) 請至少使用兩種方法 autoencoder 架構、optimizer、data preprocessing、後續降維方法、clustering 算法等等) 來改進 baseline code 的 accuracy。

a. 分別記錄改進前、後的 test accuracy 為多少。

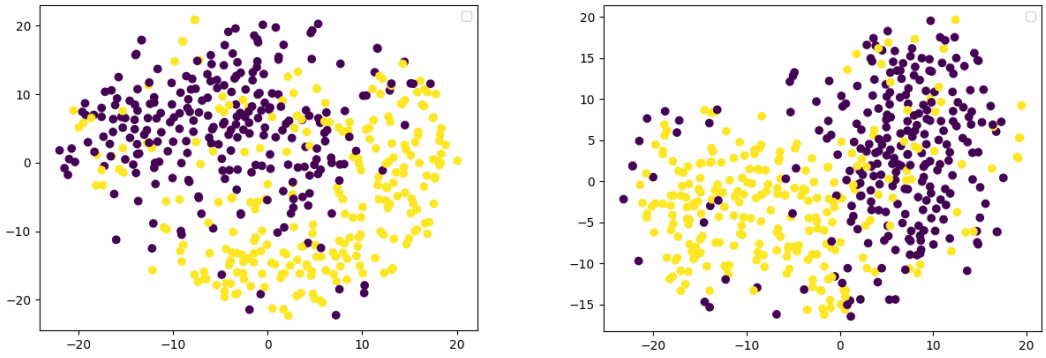
最終選擇 167epoch 為 baseline model，180epoch 為 improved=best model

	Loss on trainX / Accuracy on kaggle	
	Baseline model	Improved=best model
160 epoch	0.03973 /	0.02192 / 0.76753
167/169 epoch	0.03688 / 0.75600	0.02125 / 0.77176
180 epoch	0.03829 /	0.02052 / 0.77600
190/192 epoch	0.03757 /	0.01991 / 0.76894
200 epoch	0.03688 /	0.01956 / 0.76941

b. 分別使用改進前、後的方法，將 val data 的降維結果 (embedding) 與他們對應的 label 畫出來。

下圖，左邊是改進前的分類結果，右邊是改進後的分類結果：

可以看出，改進後的分類結果明顯有同類(色)更集中，不同類(色)更分開的趨勢。



c. 盡量詳細說明你做了哪些改進。

這次的改進主要放在第一階段的抽 latent vector 的部分，第二階段的再降維和分群則只配合第一階段的改動作一些 input 上的修正，沒有方法上的變動。首先因為是影像類的題目，所以為了避免 overfitting training set，相較於原本的 baseline model，我使用了 torchvision.transform 做了一些 data argumentation，其中包含了 RandomHorizontalFlip(), RandomRotation(15), ColorJitter(brightness = (0.5, 1.5), contrast = (0.5, 1.5), saturation = (0.5, 1.5)), RandomPerspective(), RandomAffine(15), 讓 improved model 在 accuracy 的表現上更穩定些 (見 prob3.f 圖)，而在 autoencoder 的架構上，嘗試過把它變得更 deep，三層以上，試了幾種效果都不是很好，看了一些文獻發現好像其他 autoencoder 也都是兩三層而已，因此轉而朝向把參數變多的方向嘗試，深度和原本一樣 encoder 三層 decoder 三層，但將每層的 neuron 增加，實驗結果是只增加 encoder 最後一層的 256 至 512，配合 decoder 第一層也由 256 增加至 512，效果有獲得一些提升 (架構見下圖)，而 optimizer 則沿用 Adam，learning rate 則因為 latent vector 的 dimension 變大（由 4096 維 -> 8192 維），且有做 data argumentation，用原 lr=10e-5 收斂較慢，因此改用

lr=10e-4 收斂較快速，如此訓練 200 個 epoch 得第一階段的 model。

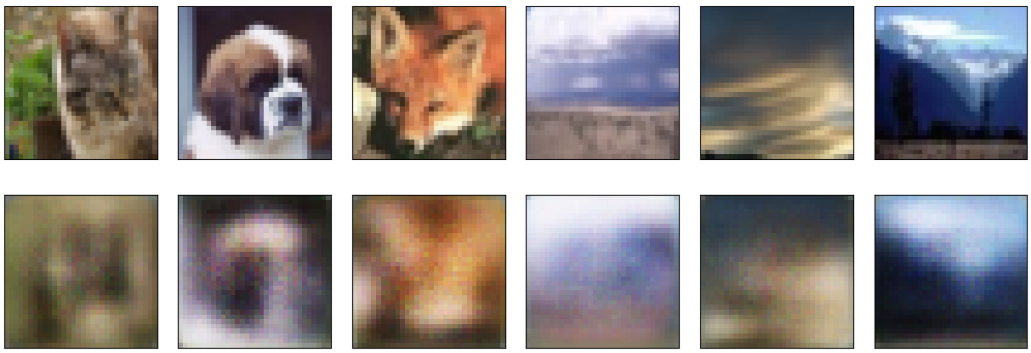
Baseline model	Improved model
(input image) 32x32x3	(input image) 32x32x3
(conv) 32x32x64 + maxPooling	(conv) 32x32x64 + maxPooling
(conv) 16x16x128 + maxPooling	(conv) 16x16x128 + maxPooling
(conv) 8x8x256 + maxPooling	(conv) 8x8x512 + maxPooling
(latent vector) 4x4x256	(latent vector) 4x4x512
(convT) 4x4x128 + unPooling	(convT) 4x4x128 + unPooling
(convT) 8x8x64 + unPooling	(convT) 8x8x64 + unPooling
(convT) 16x16x3 + unPooling	(convT) 16x16x3 + unPooling
(decoded image) 32x32x3	(decoded image) 32x32x3

而第二階段則由第一階段抽完 latent vector 後，對該 vector，經 sklearn 的 PCA 降維(至 200)和 sklearn 的 tSNE 再降維(至 2)和 sklearn 的 MiniBatchMeans 分群(至 2 類)得到最終分類的結果(accuracy)，如 prob1.a 的表和 prob1.b 的圖。

2. (1%) 使用你 test accuracy 最高的 autoencoder，從 trainX 中，取出 index 1, 2, 3, 6, 7, 9 這 6 張圖片

d. 畫出他們的原圖以及 reconstruct 之後的圖片。

上圖為 improved model 的比較結果，上排為原圖，下排為通過 autoencoder encode 再 decode 後的結果。由上圖可以看出這 autoencoder 還原圖片的能力，因為這個 improved model 是選自 180 epoch 的 weight，可以參考 prob3.b，loss 很低所以還原結果和原圖很接近，而因為 improved model 的訓練方法中有加入許多 data argumentation，所以還原出的圖邊邊角角會有一點點 distortion。而下方則附上 baseline model 的結果作比較：

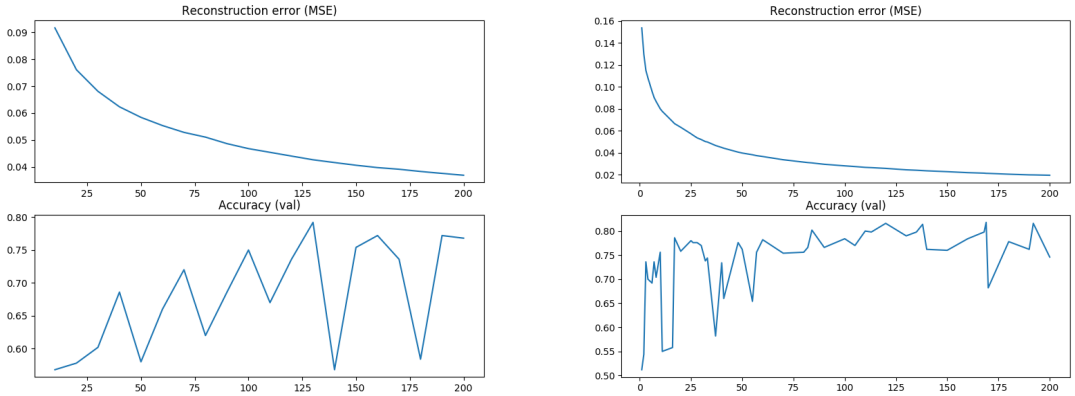


3. (2%) 在 autoencoder 的訓練過程中，至少挑選 10 個 checkpoints

e. 請用 model 的 train reconstruction error (用所有的 trainX 計算 MSE) 和 val accuracy 對那些 checkpoints 作圖。

下圖，左邊是 baseline model，右邊是 improved model：

皆為每 10 個 epoch 存一次，畫至 200epoch，但右圖還有再多存一些中間 checkpoint



f. 簡單說明你觀察到的現象。

由上圖可以發現雖然 training loss (reconstruction MSE error)會隨著 gradient descent 正常下降，也就是說通過我們設計的 auto encoder，降維後再還原出來的圖片和原圖有越來越趨近，還原度越來越高的趨勢，但是關於通過 encoder 後的 latent vector 經過一樣的 PCA 降維(至 200)和一樣的 tSNE 再降維(至 2)和一樣的 MiniBatchMeans 分群(至 2 類)的分類 accuracy，如 prob1.b 圖，並沒有隨著 loss 下降而穩定上升，反而是呈現一不太穩定的震盪。會觀察到以上結果其實是因為在這個 optimization 的問題中我們所定義的 objective function 本來就是在求通過 auto encoder 的圖要跟原圖越近越好，並不涉及經過 encoder 後的 vector 要越有分群能力越好，因此 loss 的走向並不會完全等於 accuracy 的走向，有可能是很弱的 encoder 搭配很強的 decoder 太強也可以還原出很不錯的圖片。因此若想要改善此問題，在 loss function 不變的條件下可以將 model 改為對稱的，並把 encoder 的 weight 和 decoder 的 weight 做連結，互為 transpose 關係，這樣裡面的 latent vector 就也會被連帶 train 到了，又或是可以直接改變 loss function 成也會衡量 latent vector 的好壞，這樣就會直接對 latent vector 進行 training，同時也保證 decode 出來的圖片和原圖接近。