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。

* 1. 分別記錄改進前、後的 test accuracy 為多少。

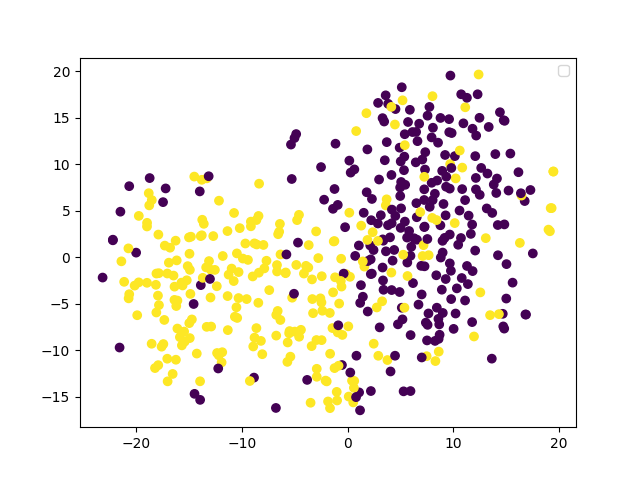
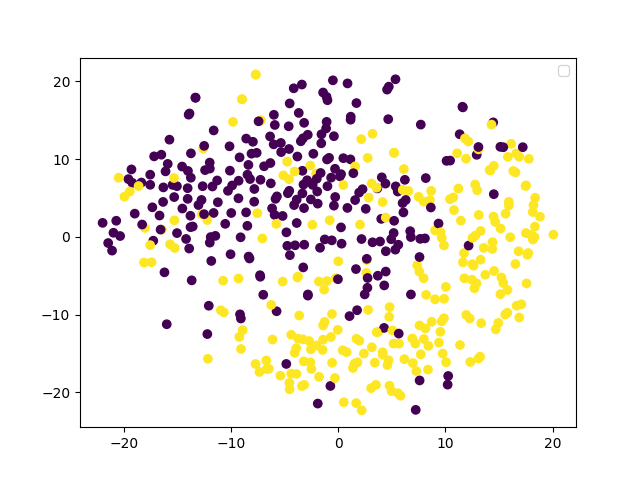
最終選擇167epoch為baseline model，180epoch為improved=best model

|  |  |  |
| --- | --- | --- |
|  | 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 |

* 1. 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。



* 1. 盡量詳細說明你做了哪些改進。

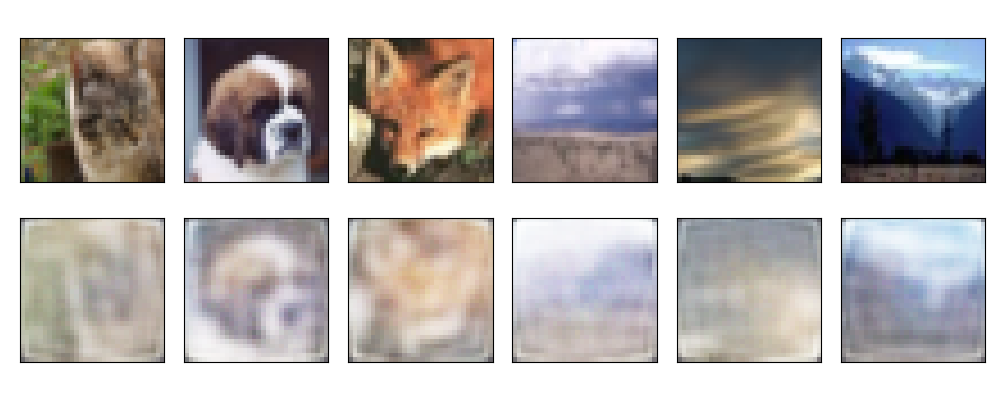
這次的改進主要放在第一階段的抽latent vector的部分，第二階段的再降維和分群則只配合第一階段的改動作一些input上的修正，沒有方法上的變動。首先因為是影像類的題目，所以為了避免overfitting training set，相較於原本的baseline model，我使用了torchvision.transform做了一些data argumentation，其中包含了RandomHorizontalFlip(), RandomRotation(15), ColorJitter(brightness = (0.5, 1.5), contrast = (0.5, 1.5), saturation = (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 |
|  |  |

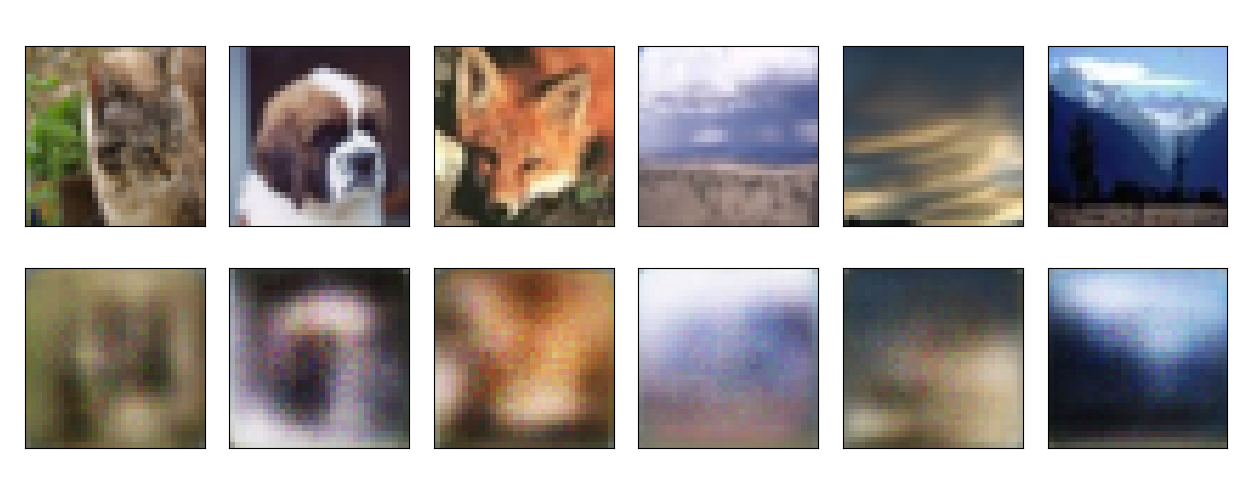
而第二階段則由第一階段抽完latent vector後，對該vector，經sklearn的PCA降維(至200)和sklearn的tSNE再降維(至2)和sklearn的MiniBatchMeans分群(至2類)得到最終分類的結果(accuracy)，如prob1.a的表和prob1.b的圖。

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

* 1. 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：

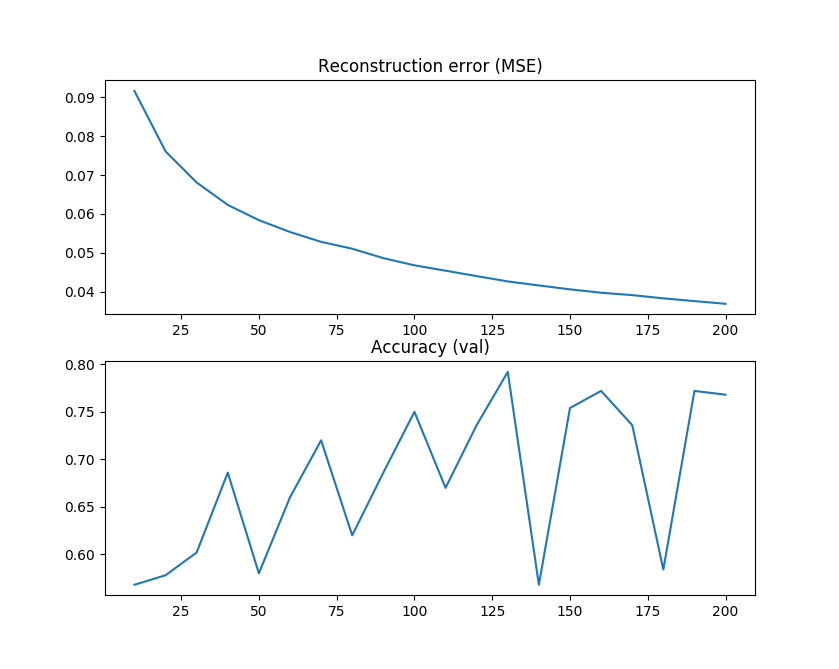
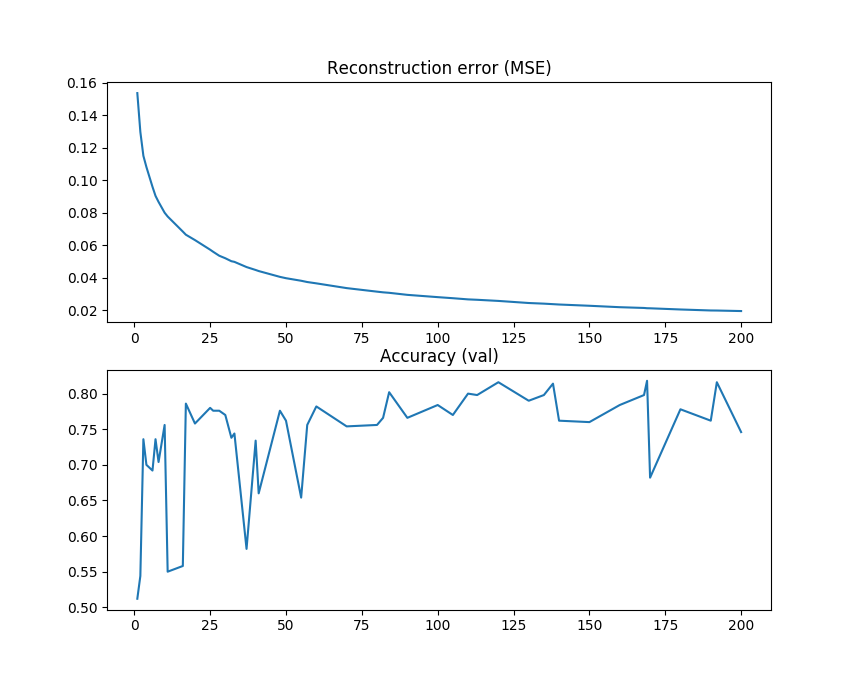


3. (2%) 在 autoencoder 的訓練過程中，至少挑選 10 個 checkpoints

* 1. 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

* 1. 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。