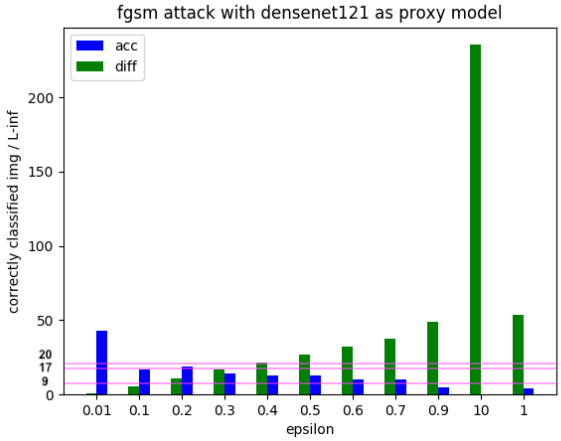
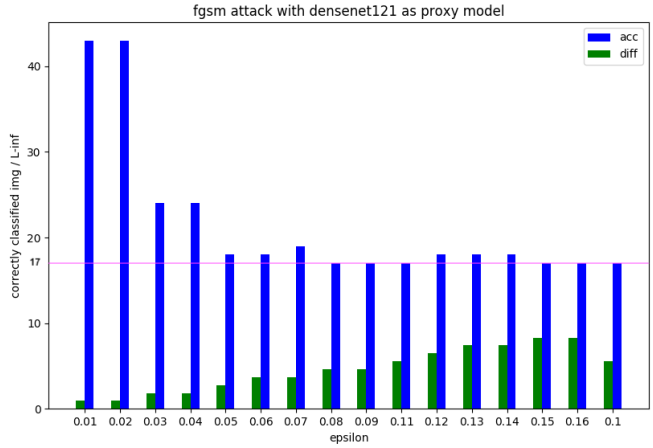
學號：r07921001　　系級：電機所二　　姓名：李尚倫

1. (2%) 試說明 hw6_best.sh 攻擊的方法，包括使用的 proxy model、方法、參數等。此方法和 FGSM 的差異為何？如何影響你的結果？請完整討論。(依內容完整度給分)

這次我們實作的是 black box 的 Adversarial Attack，其成效與方法和 proxy model 有關，其中 proxy model 選擇的部分在第二題做詳細討論，這裡直接使用結論的 pytorch pretrain 的 densenet121 當作 proxy model，而方法我們先實作了 FGSM (Fast Gradient Sign Method)的 attack，這個方法求得的方式是朝與原本 model 的 gradient decent 的反方向加減 1*epsilon，非常的簡單快速有效。而 epsilon 是個可以調整的變數，越大的話加入的雜訊越多，與原圖的 L-inf distance 也會越大，但可想而知的是會使被攻擊的 model 的 accuracy 下降越多，下圖是調整 epsilon 介於 0.01~10 之間的成果，acc 指的是 200 張被 attack 的圖片中仍然可以正確辨識的張數，越小攻擊結果越好，diff 則是指被 attack 的圖片與原圖片的 L-inf distance，越小越不容易被看出有加雜訊。

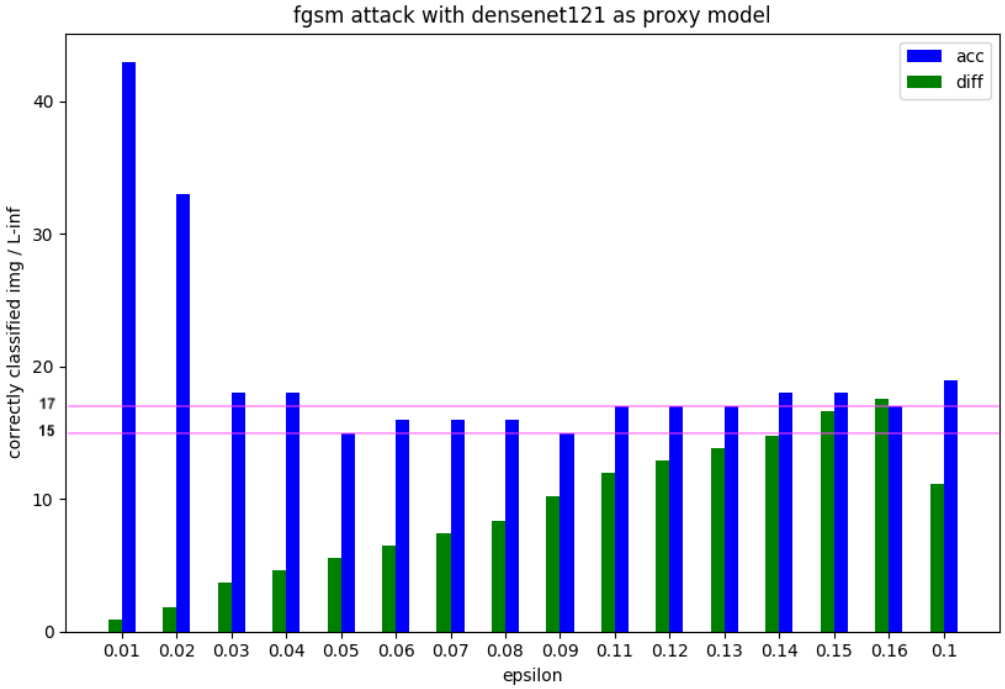


由上圖可知，epsilon 落在 0.1 的位置效果是最好的，acc=17，正確辨識率= 17/200 = 8.5%，攻擊成功率 = 91.5%，diff=5.55，幾乎已經過 strong baseline 了，而若在 0.1 附近再套用不同的 epsilon 看更細微的差異，則如下圖，



可以發現 acc 無法再更好，但 diff 可以再往下壓到 epsilon=0.08/0.07，diff=4.625。但經

由這次實驗也發現，epsilon 在大趨勢上是越大 acc 越低 diff 越大，但在小範圍內則不一定，原因可能是出在再加入雜訊的過程中，可能已經超過圖片 0~255 的區間因此被 clip 掉，導致無法反應原本預期要餵入 model 的值。因此可以再嘗試在這個區間內使用不同的方式去 improve，其中想到的一個就是在 sign 不同時給不同的 weight，-1 時乘上 1.0 不變，而+1 時則調整乘上不同的 weight，讓加入的雜訊根據 gradient 的正負做不對稱的增減，直覺上的理解，原本的 model 是，如果這個區塊很像目標，就變得不要這麼像，如果這個區塊很不像目標，就變得不要這麼不像。而現在這個方法就是讓如果這個區塊很像目標，就要變得很不像，加強力道。結果如下圖，weight = 2.0。



可以發現 acc 成功下降到了 15，正確辨識率= 15/200 = 7.5%，攻擊成功率 = 92.5%，diff=5.55 (但在測試平台上是 2.7750，尚不清楚原因)，成功超越 strong baseline，而若改變 weight 在 1.0~2.0 的區間做變動，效果皆差不多，最好就是做到 acc=15。

2. (1%) 請嘗試不同的 proxy model，依照你的實作的結果來看，背後的 black box 最有可能為哪一個模型？請說明你的觀察和理由。

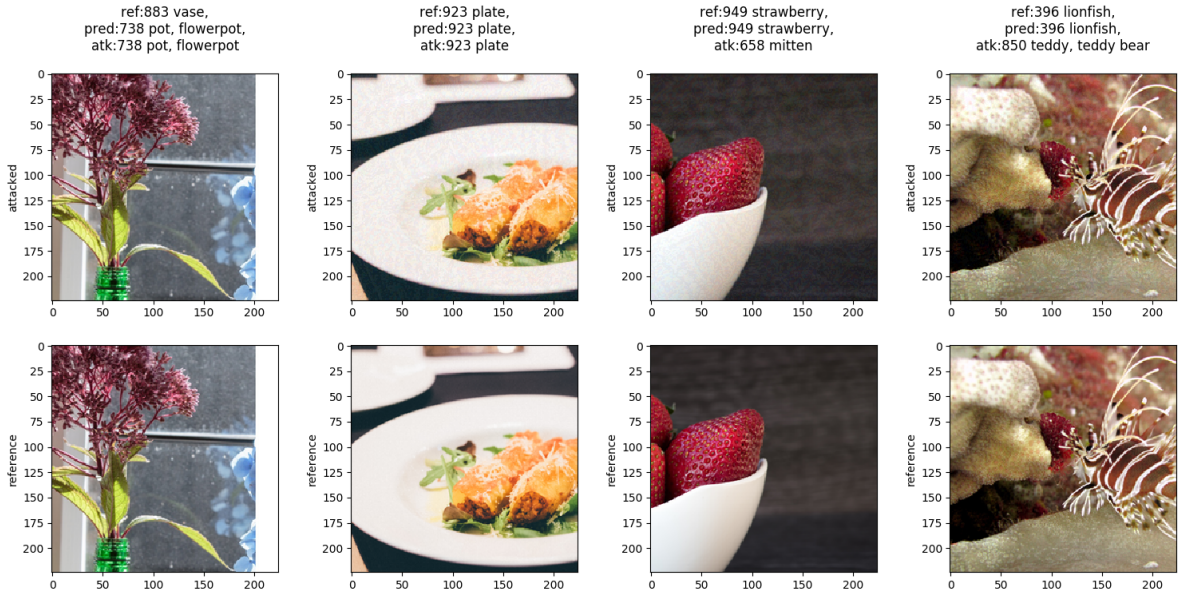
因為這次實作的是 black box 的 Adversarial Attack，其成效與方法和 proxy model 有很大的關係，因此必須要來測試基於同一個攻擊方法(FGSM, epsilon=0.4)哪個 proxy model 的成效最好，而下方表格則是我在 vgg16, vgg19, resnet50, resnet101, densenet121, densenet169 在 pytorch 上的 pretrain model 的測試結果，而這裡的 accuracy 則是指攻擊成功的圖片/總圖片個數(200 張)*100%， local accuracy 則是使用同樣該 model 去測試的結果，online accuracy 則是使用功課網站未知 model 的測試結果。

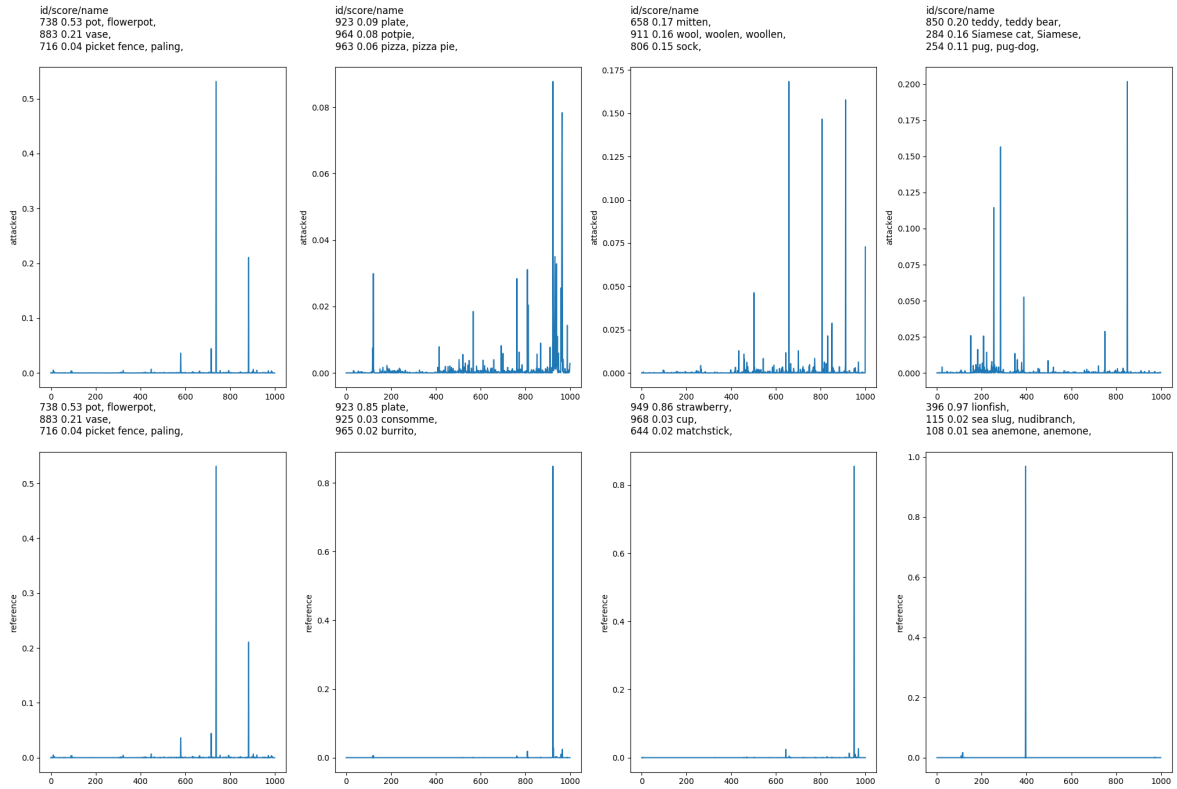
Model	Local accuracy	Online accuracy	L-inf
Vgg16	0.96	0.565	19.895
Vgg19	0.955	0.55	20.01
Resnet50	0.91	0.72	23
Resnet101	0.87	0.62	21.39
Densenet121	0.935	0.935	21.275
Densenet169	0.87	0.74	21.045

由上面表格可以很明顯地看出，如果使用的 proxy model 和黑箱的 model 越接近的話攻擊的效果會越好，因為我們的方法是根據 gradient 的方向來做出 noise，因此不同的 model 就會做出不同的適應該 model 的 noise，因此是非常 depend on model 的，由結果看來最接近的就是 densenet121 (不只接近，根本就一樣)，因此後續都是使用 densenet121 來當 proxy model 進行 Adversarial Attack。

3. (1%) 請以 hw6_best.sh 的方法，visualize 任意三張圖片攻擊前後的機率圖 (分別取前三高的機率)。

由下圖實驗可以看到，我分別取了四張不同情況下的攻擊前後成果，從左數來第一張，因為原本的 proxy model predict 的結果就和 ground truth 不符，所以並沒有加入 attack 的雜訊，分類的 distribution 和原本一致，ground truth 的 score 為 0.21。第二張，是 attack 失敗的情況，雖然失敗，但我們可以看到分類的 distribution 上確實有被擾亂到，原本集中的預測結果變得非常分散，只是結果上 plate 的 score 仍然是最高，但已由原本的 score=0.85 降到 score=0.09，並和其他前兩名的類別非常接近。第三、四張是 attack 成功的結果，我們可以看到分類的 distribution 上確實有被擾亂到，原本集中的預測結果變得非常分散，前三名高的分類和原本的完全不一樣，很成功。





4. (2%) 請將你產生出來的 adversarial img，以任一種 smoothing 的方式實作被動防禦 (passive defense)，觀察是否有效降低模型的誤判的比例。請說明你的方法，附上你防禦前後的 success rate，並簡要說明你的觀察。另外也請討論此防禦對原始圖片會有什麼影響。

以下是我防禦的 pipeline：

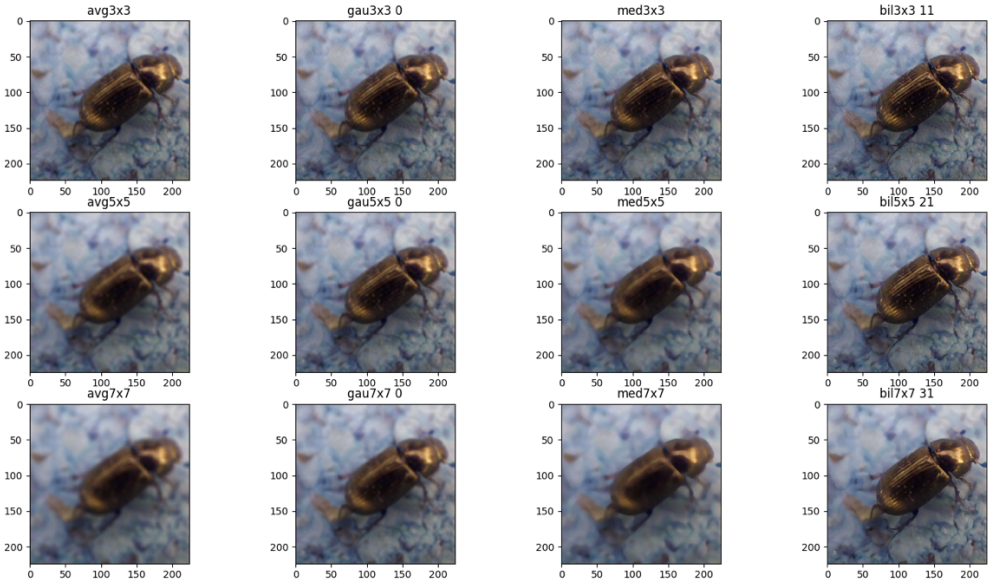
原始圖 → 防禦 smoothing → 進 proxy model → prediction

→進 attack model → 進 proxy model → prediction

Method	Orig acc	Atked acc	Blur orig acc	Blur atked acc
Averaging blur3x3	**92.5**	**7.5**	**0.86**	**0.26**
Averaging blur5x5			0.76	0.39
Averaging blur7x7			0.61	0.45
Gaussian blur 3x3			**0.855**	**0.195**
Gaussian blur 5x5			0.84	0.285
Gaussian blur 7x7			0.74	0.37
Median blur 3x3			**0.905**	**0.28**
Median blur 5x5			0.755	0.385
Median blur 7x7			0.6	0.395
Bilateral filter 3x3, 11			0.915	0.075
Bilateral filter 5x5, 22			0.905	0.075
Bilateral filter 7x7, 33			**0.89**	**0.14**

由上表結果可以看出，不管做哪一種 blur，都會對辨識結果造成影響，越強的防禦效果，往往導致正常的辨識結果也大受影響，其中我們比較了四種 blurring 的方式，分別是 opencv 內建的 gaussian blur, median blur, average blur，其中我們發現 median blur 的效果是全部裡面最好的，保有最高的正常辨識成功率的同時又能有不錯的防禦力，在被 atk 的情況下準確率有比沒防禦前進步了四倍。但若兩要兩者兼顧，正常的圖片和被攻擊過的圖片都要有不錯的辨識成功率的話，就 passive 的防禦方式來說，還是需要一些機制的配合，像是上課提到的 Feature Squeeze 的方式，應該就能改善這個問題。而下方則列出被攻擊後的圖片加上各個 blur 效果在不同 window size 下的效果，和沒被攻擊的圖片加上各個 blur 效果在不同 window size 下的效果

defense: atked_img_blur



defense: orig_img_blur