Machine Learning HW5 Report

學號：B06902049 系級：資工二 姓名：林首志

1. (1%) 試說明 hw5_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)

   答：我使用的proxy model是torchvision的resnet50，模型參數是torchvision提供的pretrained參數。

   我的攻擊演算法是修改過的DeepFool (https://arxiv.org/abs/1511.04599)。

   每次讀進一張圖片時，我先對圖片做預處理，讓其符合pytorch pretrained model的輸入格式。

   之後就會開始跑DeepFool的演算法，迭代次數最多50次。在開始迭代前我會先用model對原圖做一次預測，記錄機率前10高的classes。每次迭代時，如原論文中提到的ℓ_∞ norm multiclass的作法，我會取出$\hat{l} = \arg\min_{k \neq \hat{k}(\mathbf{x}_0)} \frac{|f'_k|}{||\mathbf{w}'_k||_1}$（其中$k$只包含機率前10高的classes），並令$\mathbf{r}_i = \frac{|f'_{\hat{l}}|}{||w'_{\hat{l}}||_1}\text{sign}(\mathbf{w}'_{\hat{l}})$，對圖片$\mathbf{x}_i$做$\mathbf{x}_{i+1} = \mathbf{x}_i + (1+0.05)\mathbf{r}_i$的更新。更新完後，我會對圖片做clip的操作，確保圖片的ℓ_∞ norm在目標範圍內。clip操作的ϵ我設為$2/255/0.229$（讓最終的L-inf norm大約落在2左右，由於預處理會將數值範圍轉成[0, 1]之後再分別對每個channel做標準化，我將eps除以255，再除以其中最大的標準差0.229）。

   每次迭代之前，我會檢查目前的圖片是否已經能成功fool classifier。如果可以，我會檢查圖片的ℓ_∞ norm是不是大於等於$\epsilon/2$（ϵ即是clip操作的ϵ），如果是，則會提早結束DeepFool的迭代；如果否，我會對圖片做$\mathbf{x}_{i+1} = \frac{\epsilon}{2} \times \frac{\mathbf{r}_{i-1}}{\max \mathbf{r}_{i-1}}$（$\max \mathbf{r}_{i-1}$代表$\mathbf{r}_{i-1}$的elements中的最大值）的更新，經過clip後直接進行下一次的迭代。這樣做是為了確保圖片最後轉為整數儲存時，不會因為原先的ℓ_∞太小而造成儲存後的圖片沒辦法fool classifier的問題。

   DeepFool的演算法結束後，我會對圖片做postprocess（大略為預處理的逆操作），才將圖片儲存下來。

   此方法和FGSM的差異是，(Non-targeted) FGSM的更新方向是由原類別的loss的梯度方向所決定的，而且只更新一次，步伐大小只能由給定的ϵ控制。而DeepFool的更新方向大致上是由往最近的其他類別的垂直方向所決定的，而且更新次數比較多，更新的步伐大小也比較精準。FGSM只考慮原類別，而DeepFool不但多考慮了其他類別，迭代次數和步伐大小的精準度都比較高，因此DeepFool的結果比較好，可以用更少的擾動來達到更高的成功率。但由於FGSM的運算量比較小，計算速度比DeepFool還要快，FGSM還是有他的優勢。

2. (1%) 請列出 hw5_fgsm.sh 和 hw5_best.sh 的結果 (使用的 proxy model、success rate、L-inf. norm)。

   答：結果如下表，可見Best的效果的確比較好。

	FGSM	**Best**
proxy model	Pytorch pretrained ResNet50	Pytorch pretrained ResNet50
success rate	0.895	0.995
L-inf. norm	3.0000	2.0050

3. (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。

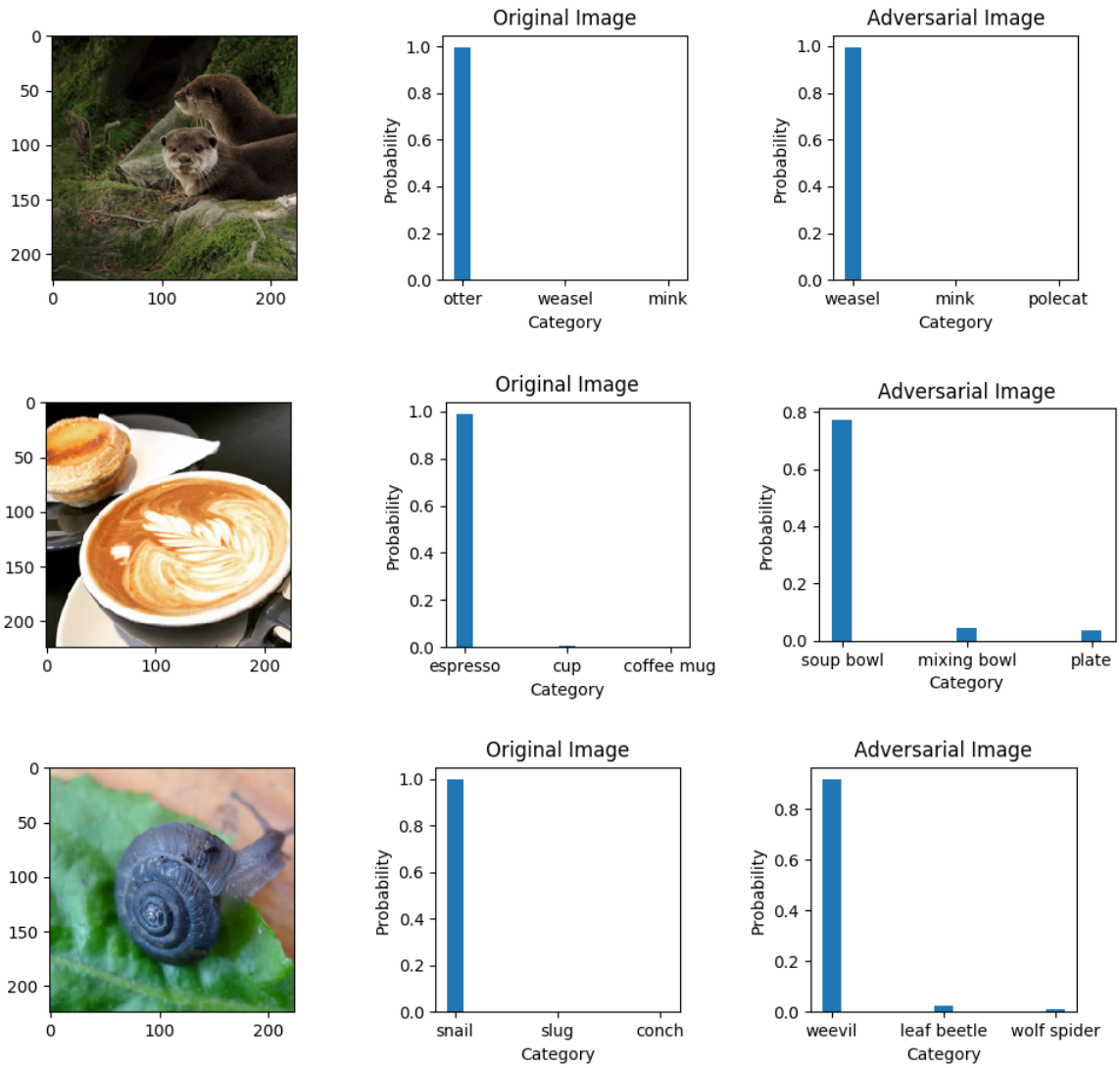
   答：我將同樣的演算法(Best)套用在不同的proxy model上，結果如下：（下列都是pytorch的pretrained model）

Proxy Model	Success Rate	L-inf. norm
VGG16	0.110	2.0050
VGG19	0.105	2.0050
ResNet50	0.995	2.0050
ResNet101	0.260	2.0000
DenseNet121	0.150	2.0000
DenseNet169	0.155	2.0050

根據結果，背後的black box model最有可能是ResNet50，因為使用他攻擊的成功率遠遠超過其他models，而且其成功率和我在本機用Pytorch pretrained resnet50測試出來的成功率是一模一樣的。

4. (1%) 請以 hw5_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)。

答：以下是047.png、088.png、185.png的結果：（由於pytorch的model沒有softmax layer，機率的算法是model的輸出手動套用softmax的結果）



對於這三張圖片，ResNet50對原始圖片都能很有信心的預測出正確的結果，但是遇到Adversarial Images時卻都很有信心的預測出一個錯誤的答案。由於我的演算法只針對機率前10高的類別做攻擊，可以發現Adversarial Images預測出的類別和原始類別有一定程度的相似。

5. (1%) 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。

   答：我將產生出來的adversarial images用opencv讀進來，套用kernel size=(3, 3), sigma為opencv預設值的GaussianBlur，再用opencv將處理完的圖片儲存起來。下表是防禦前後的success rate。

	Success rate
防禦前	0.995
防禦後	0.680

   可以發現GaussianBlur的確能降低誤判的比例，如果套用更強的模糊也許效果會更好。

   下表是此防禦對原始圖片的影響：

	Success rate
防禦前	0.000
防禦後	0.115

   可以發現雖然防禦降低了誤判Adversarial images的比例，但卻增加了誤判原始圖片的機率，可見此防禦並不是非常理想的方法。