Machine Learning HW5 Report

學號：B05902018　　系級：資工三　　姓名：張凱程

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)

Ans: 使用方法為 Basic Iterative Method，proxy model 是 resnet50，參數為 steps = 40, learning rate = 3e-4，資料有做 normalization，此方法跟 FGSM 的差異為 FGSM 只做一次，即 steps = 1，BIM 可以做很多次。FGSM 的攻擊結果理論上為原圖 +- learning rate，而因為 BIM 做比較多次，攻擊結果不一定，但為了讓攻擊強度夠且 L-inf. Norm 要夠少，所以要利用反覆攻擊的模式確保攻擊成功，也因為多次攻擊的關係，攻擊結果不一定為原圖 +- learning rate，增加了變因，攻擊也相對容易成功。

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)。

Ans:

FGSM: proxy model = resnet50, success rate = 0.815, L-inf. Norm = 6.0000

BEST: proxy model = resnet50, success rate = 0.970, L-inf. Norm = 1.0000

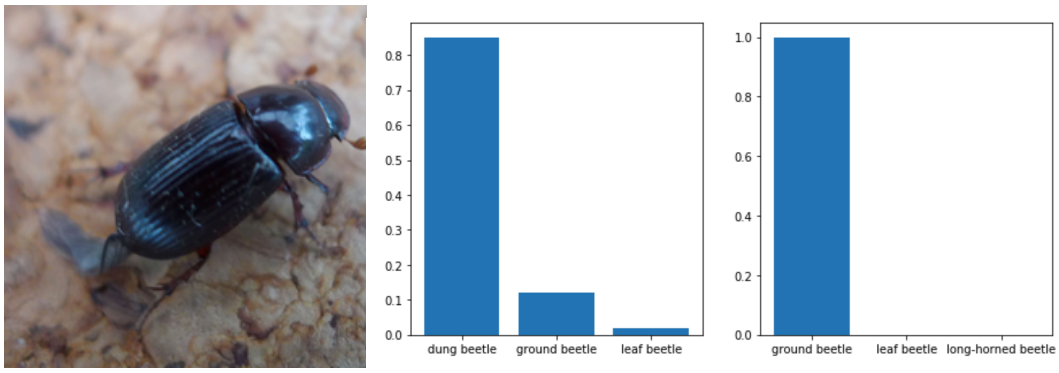
3. (1%) 請嘗試不同的 proxy model，依照你的實作的結果來看，背後的 black box 最有可能為哪一個模型？請說明你的觀察和理由。

Ans:

應為 resnet50，其他模型做相同作法時(以 best 為例)，正確率都在 0.2XX，而 Densenet121 有特別把沒攻擊成功的或不影響結果的換成原圖，攻擊結果反而更低，顯示出 Densenet121 攻擊失敗的結果在 block box 反而成功，顯然是猜測錯誤。

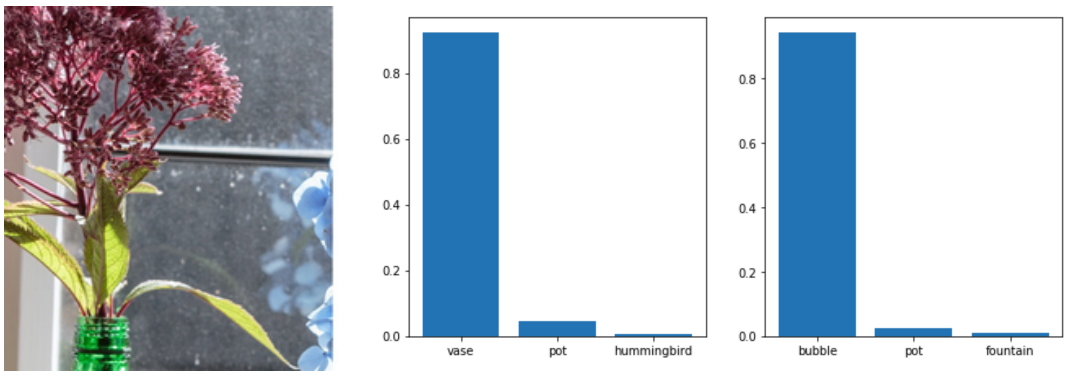
4. (1%) 請以 hw5_best.sh 的方法，visualize 任意三張圖片攻擊前後的機率圖 (分別取前三高的機率)。

Ans:



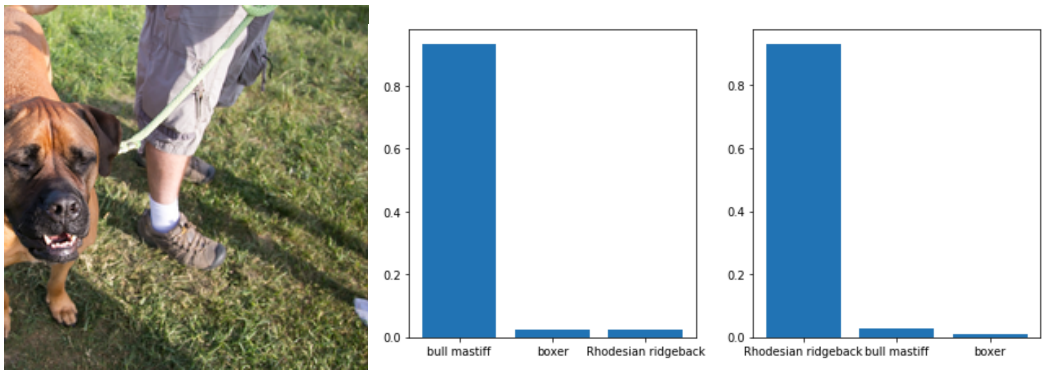
Before: dung beetle 0.8502659

After: ground beetle, carabid beetle 0.99857104



Before: vase 0.9235093

After: bubble 0.9421538



Before: bull mastiff 0.9335059

After: Rhodesian ridgeback 0.93151164

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。

Ans:

以下是沒做 normalization 的結果：

使用方法：Gaussian Filter (Use Scipy, alpha = 0.5)

Before filtering: success rate = 0.955, L-inf. Norm = 5.0000

After filtering: success rate = 0.945, L-inf. Norm = 25.0450

觀察：alpha 很小的時候，防禦完全沒有效果，alpha 很大的時候，正確率也沒有明顯的下降，反而 L-inf. Norm 大幅提升，也就是說 smoothing 可能沒有很有效，但可以讓原 model 擁有者比較容易偵測到攻擊後的圖像。

以下是有做 normalization 的結果：

使用方法：Pillow 的 .filter(ImageFilter.SMOOTH)

Before filtering: success rate = 0.970, L-inf. Norm = 1.0000

After filtering: success rate = 0.480, L-inf. Norm = 75.8250

觀察：

不僅 L-inf. Norm 增加，正確率也明顯下降，可以看出做 normalization 不管做攻擊或作防禦都比較有效。

對原圖作 smoothing：

使用方法：Pillow 的 .filter(ImageFilter.SMOOTH)

success rate = 0.080, L-inf. Norm = 75.5450

觀察：做 smoothing 會讓比較銳利的 pixel 變平滑，此 pixel 差異會比較大，也造成 L-inf. Norm 變大。