

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)


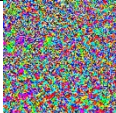

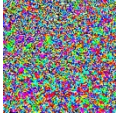



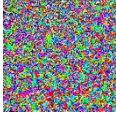



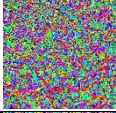

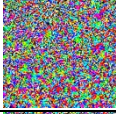

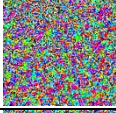

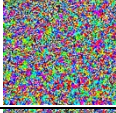


Proxy model: Densenet121。

方法: iteratively 修改圖片，每次都把圖片加上 0.001 乘上 gradient。

參數：epsilon 為每次迭代 0.001、超過 100 次就取消攻擊（結果是都有成功）

差異：FGSM 是一次性直接改，這個則是重複加上當前的 gradient，

結果：FGSM 只能到大概 6 成，iterative 可以到 9 成

Result	Original	Gradient	Prediction1	Prediction2	Prediction3
Fail			bull mastif 15.63	boxer 11.79	Great Dane 11.60
Fail			bull mastif 15.18	boxer 11.72	Great Dane 11.64
Fail			bull mastif 14.76	Great Dane 11.69	boxer 11.64
Fail			bull mastif 14.36	Great Dane 11.75	boxer 11.56
Fail			bull mastif 13.97	Great Dane 11.85	boxer 11.46
Fail			bull mastif 13.57	Great Dane 11.92	boxer 11.35
Fail			bull mastif 13.19	Great Dane 12.02	boxer 11.23
Fail			bull mastif 12.83	Great Dane 12.83	Rhodesian ridgeback 12.15
Fail			bull mastif 12.47	Great Dane 12.29	Rhodesian ridgeback 12.29
Success			Great Dane 12.45	bull mastif 12.13	Rhodesian ridgeback 12.13

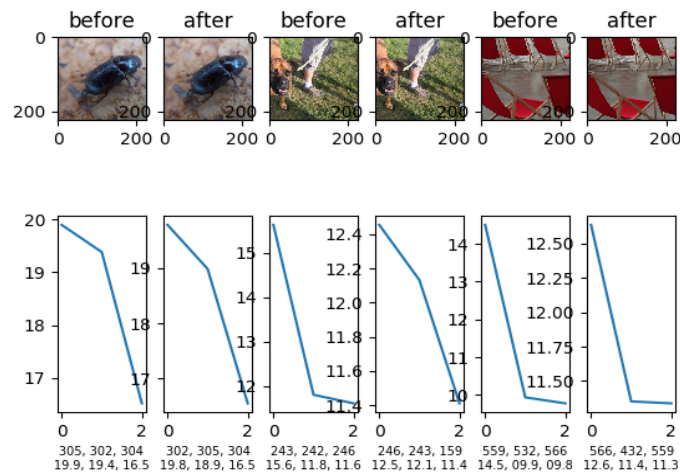
2. (1%) 請嘗試不同的 proxy model，依照你的實作的結果來看，背後的 black box 最有可能為哪一個模型？請說明你的觀察和理由。

以一模一樣的 code 換幾個可能的 model 後得到：

Success Rate on JudgeBoi	VGG-16	VGG-19	ResNet-50	ResNet-101	DenseNet-121	DenseNet-169
	0.085	0.090	0.120	0.110	0.955	0.135

由上面的表格可以合理的猜測，black box 為 DenseNet-121。

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)。



Index		Prediction1	Prediction2	Prediction3
0	Before	dung beetle	ground beetle	leaf beetle
		19.898052	19.383718	16.517605
	After	ground beetle	dung beetle	leaf beetle
		19.78683	18.98768	16.549694
2	Before	bull mastiff	boxer	Great Dane
		15.6293955	11.798059	11.604064
	After	Great Dane	bull mastiff	Rhodesian ridgeback
		12.45495	12.132084	11.410471
3	Before	folding chair	dining table	French horn
		14.505009	9.936819	9.778683
	After	French horn	bassoon	folding chair
		12.63637	11.351826	11.3375435

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。

方法：Randomize at Inference Phase

只需改一行，便是把 `torchvision.transforms.Resize((224, 224), interpolation=3)` 改成 `torchvision.transforms.RandomResizedCrop(224)`，也就是從將原圖整個 **resize** 成(224, 224)，變成隨機地選取原圖中的某個位置，**attack success rate** 可以直接從 0.955 變成 0.4。

至於為什麼這樣做可以降低被攻擊的機率，是因為攻擊通常是針對幾個特定的 **pixel** 進行的，在 **predict** 前若先隨機 **resize**、**padding** 等等，原先可以拿來攻擊的位置都會被改掉，也因而可以成功防禦了。

不過這種防禦方式需要注意的是，如果把圖片改得太誇張，可能會導致模型反而完全無法判斷了，所以在使用的時候要小心，才能達到最佳的防禦效果。

對原始圖片的影響：隨機 **crop** 某個範圍跟 **resize** 圖片。