

Machine Learning HW5 Report

學號：B05901040 系級：電機三 姓名：蔡松達

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。

此處使用的 proxy model 是 VGG19 (include_top=True, weights='imagenet'，使用 keras 預先 train 好的 weight 沒有做任何調整)，使用的方法雖然仍是 FGSM，但是與原先的 FGSM 的差別在於預處理，我試了兩種預處理的方式，一種是 caffe mode，另一種是 torch mode，處理的方式如下：

mode	method	
caffe	preprocess_input	將圖片從 RGB 轉換至 BGR，並分別減去 ImageNet dataset 的平均 BGR [103.939, 116.779, 123.68]
	output (轉換回來)	將圖片從 BGR 轉回至 RGB，並分別減去 ImageNet dataset 的平均 RGB[123.68, 116.779, 103.939]
torch	preprocess_input	所有 pixel 除以 255，減去 ImageNet 平均[0.485, 0.456, 0.406]，除以標準差[0.229, 0.224, 0.225]
	output (轉換回來)	所有 pixel 乘上標準差[0.229, 0.224, 0.225]，加上 ImageNet 平均[0.485, 0.456, 0.406]，再乘上 255

hw5_fgsm 是使用 torch 的方式轉換，hw5_best 的方式是使用 caffe mode，兩者調整的參數不同，hw5_fgsm 使用 epochs = 1 & epsilon = 0.35；hw5_best 則是使用 epochs = 1 & epsilon = 4.7，兩者的結果列在第二題中，雖然以 success rate 來說前者較好，但是 L-inf. norm 大出不少，對此我做了另一個實驗 (mode = "torch", epochs = 1 & epsilon = 0.1) 在 hw5_fgsm 上，但是得到的結果則如第二題所示，也就是使用 torch 時不僅 L 會變大且 success rate 會降低，只有在使用較大 epsilon 時才能通過 simple baseline；但 hw5_best 採用 caffe mode，不僅皆能通過 simple baseline，L-inf. Norm 更是能通過 strong baseline，效果較 torch mode 來的好。這樣的結果其實也算是顯而易見，因為 keras 的 VGG19 在 import preprocessing_input 時就是使用 caffe mode 的方式，應該也是因為 VGG19 的架構比較適用在這樣的預處理架構上，因此呈現出的結果也較好。

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)。

所有 VGG19 皆使用 keras 預先 train 好的 weight 沒有做任何調整(include_top=True, weights='imagenet')，VGG19 架構則如下圖所示：

	proxy model	success rate	L-inf. norm
hw5_fgsm.sh (epochs = 1 & epsilon = 0.35)	VGG19 with preprocess mode = torch	0.57 (simple)	21.00 (simple)
hw5_best.sh (epochs = 1 & epsilon = 4.7)	VGG19 with preprocess mode = caffe	0.4(simple)	4.80(strong)

epsilon = 4.7)			
hw5_fgsm.sh 改成 使用 epochs = 1 & epsilon = 0.1(此處 只是用來比較)	VGG19 with preprocess mode = torch	0.17 (fail)	6.00 (simple)

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	(None, 224, 224, 3)	0
block1_conv1 (Conv2D)	(None, 224, 224, 64)	1792
block1_conv2 (Conv2D)	(None, 224, 224, 64)	36928
block1_pool (MaxPooling2D)	(None, 112, 112, 64)	0
block2_conv1 (Conv2D)	(None, 112, 112, 128)	73856
block2_conv2 (Conv2D)	(None, 112, 112, 128)	147584
block2_pool (MaxPooling2D)	(None, 56, 56, 128)	0
block3_conv1 (Conv2D)	(None, 56, 56, 256)	295168
block3_conv2 (Conv2D)	(None, 56, 56, 256)	590080
block3_conv3 (Conv2D)	(None, 56, 56, 256)	590080
block3_conv4 (Conv2D)	(None, 56, 56, 256)	590080
block3_pool (MaxPooling2D)	(None, 28, 28, 256)	0
block4_conv1 (Conv2D)	(None, 28, 28, 512)	1180160
block4_conv2 (Conv2D)	(None, 28, 28, 512)	2359808
block4_conv3 (Conv2D)	(None, 28, 28, 512)	2359808
block4_conv4 (Conv2D)	(None, 28, 28, 512)	2359808
block4_pool (MaxPooling2D)	(None, 14, 14, 512)	0
block5_conv1 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv2 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv3 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv4 (Conv2D)	(None, 14, 14, 512)	2359808
block5_pool (MaxPooling2D)	(None, 7, 7, 512)	0
flatten (Flatten)	(None, 25088)	0
fc1 (Dense)	(None, 4096)	102764544
fc2 (Dense)	(None, 4096)	16781312
predictions (Dense)	(None, 1000)	4097000
Total params: 143,667,240		
Trainable params: 143,667,240		
Non-trainable params: 0		

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。

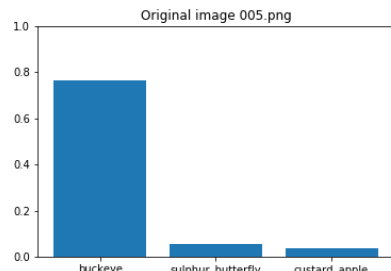
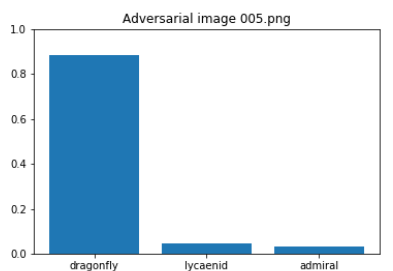
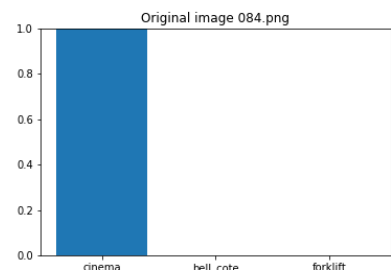
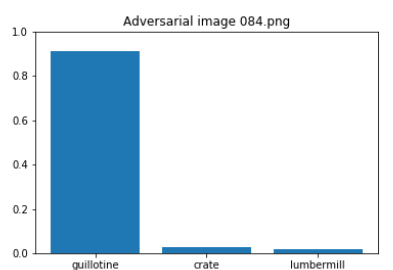
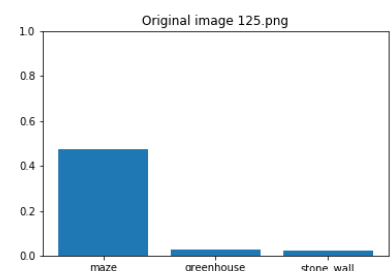
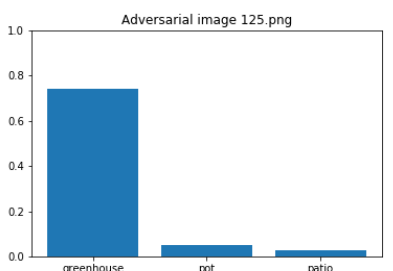
此處取前四十張照片並使用六種 model 進行 attack（其餘 160 張就直接上傳不攻擊的照片），VGG16、VGG19、Resnet50、Resnet101 使用 preprocess input: caffe mode, epochs = 1, epsilon = 20；Densenet121、Densenet169 使用 preprocess input: torch mode, epochs = 1, epsilon = 0.3，所有的 model 皆採用 pretrain 的 weight，結果如下：（success rate 最多為 0.2 因為只有對 40 張照片 attack）

proxy model	success rate	L-inf. norm
VGG16	0.165	4.2

VGG19	0.18	4.2
ResNet50	0.165	4.1
ResNet101	0.17	4.2
DenseNet121	0.14	3.6
DenseNet169	0.15	3.515

透過實驗觀察 VGG19 較 VGG16 接近；Resnet50、Resnet101 結果都不比 VGG19 來的好，DenseNet121、DenseNet169 的 L 較小，但若以比例比較，VGG19 仍舊能夠達到略好一點的結果，但由於此處使用的皆為原本的 pretrained 模型，可能需要藉由調整 weight 才能找出更精確的結果，透過此處數據觀察 VGG19 應該較為接近 black box。

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)。

圖片編號	Original	Adversarial																
005.png	<p>Original image 005.png</p>  <table><thead><tr><th>Category</th><th>Probability</th></tr></thead><tbody><tr><td>buckeye</td><td>0.77</td></tr><tr><td>sulphur_butterfly</td><td>0.05</td></tr><tr><td>custard_apple</td><td>0.03</td></tr></tbody></table>	Category	Probability	buckeye	0.77	sulphur_butterfly	0.05	custard_apple	0.03	<p>Adversarial image 005.png</p>  <table><thead><tr><th>Category</th><th>Probability</th></tr></thead><tbody><tr><td>dragonfly</td><td>0.90</td></tr><tr><td>lycaenid</td><td>0.05</td></tr><tr><td>admiral</td><td>0.03</td></tr></tbody></table>	Category	Probability	dragonfly	0.90	lycaenid	0.05	admiral	0.03
Category	Probability																	
buckeye	0.77																	
sulphur_butterfly	0.05																	
custard_apple	0.03																	
Category	Probability																	
dragonfly	0.90																	
lycaenid	0.05																	
admiral	0.03																	
084.png	<p>Original image 084.png</p>  <table><thead><tr><th>Category</th><th>Probability</th></tr></thead><tbody><tr><td>cinema</td><td>1.00</td></tr><tr><td>bell_cote</td><td>0.00</td></tr><tr><td>forklift</td><td>0.00</td></tr></tbody></table>	Category	Probability	cinema	1.00	bell_cote	0.00	forklift	0.00	<p>Adversarial image 084.png</p>  <table><thead><tr><th>Category</th><th>Probability</th></tr></thead><tbody><tr><td>guillotine</td><td>0.92</td></tr><tr><td>crate</td><td>0.03</td></tr><tr><td>lumbermill</td><td>0.02</td></tr></tbody></table>	Category	Probability	guillotine	0.92	crate	0.03	lumbermill	0.02
Category	Probability																	
cinema	1.00																	
bell_cote	0.00																	
forklift	0.00																	
Category	Probability																	
guillotine	0.92																	
crate	0.03																	
lumbermill	0.02																	
125.png	<p>Original image 125.png</p>  <table><thead><tr><th>Category</th><th>Probability</th></tr></thead><tbody><tr><td>maze</td><td>0.48</td></tr><tr><td>greenhouse</td><td>0.02</td></tr><tr><td>stone_wall</td><td>0.01</td></tr></tbody></table>	Category	Probability	maze	0.48	greenhouse	0.02	stone_wall	0.01	<p>Adversarial image 125.png</p>  <table><thead><tr><th>Category</th><th>Probability</th></tr></thead><tbody><tr><td>greenhouse</td><td>0.75</td></tr><tr><td>pot</td><td>0.05</td></tr><tr><td>patio</td><td>0.02</td></tr></tbody></table>	Category	Probability	greenhouse	0.75	pot	0.05	patio	0.02
Category	Probability																	
maze	0.48																	
greenhouse	0.02																	
stone_wall	0.01																	
Category	Probability																	
greenhouse	0.75																	
pot	0.05																	
patio	0.02																	

這些圖在經過 adversarial attack 之後都得到完全不同的結果。

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。

此處使用 Gaussian Filter 進行 smoothing，使用近似 3*3 Gaussian Filter 的 generalized weighted smoothing filter 矩陣，其 convolution 的 kernel 如下圖所示：

0.0449192	0.122103	0.0449192
0.122103	0.331911	0.122103
0.0449192	0.122103	0.0449192

使用此 kernel 對圖片進行 convolution 後可以得到 smoothing 最終的結果(周邊點因為無法進行 convolution 故補 0)，而藉由下圖例子可以看出 smoothing 過後的圖片變得較模糊，success rate 由原先的 0.57 降為 0.475，並沒有非常顯著的效果，可能的原因是因為在模糊的過程中 L-inf norm 也從 21 上升至 78.69，說明圖片被改變的幅度變大，但是 success rate 仍舊有所下降，可見 smoothing 對於 defense 仍舊有一定的效果，也可能是這個 kernel 選的不佳，需要再嘗試更多種 kernel，來達到最好的 defense 的效果。此外下面也放上不經 attack 直接經過 Gaussian Filter 的圖片，可以看得出與原圖較為接近，只是模糊了些，其成功率也是相當低只有 0.08，不過 L-inf norm 卻來到 77.015，可見雖然圖片被大幅度的變動，卻幾乎仍舊能夠完全預測正確，表示經過 Gaussian Filter 對於圖片主要是防禦而非攻擊的效果。



經過 attack 的圖片
Success rate: 0.57



再經過 Gaussian Filter 的圖片
Success rate: 0.475



未經過 attack 就經過 Gaussian Filter 的圖片
Success rate: 0.08



原圖