

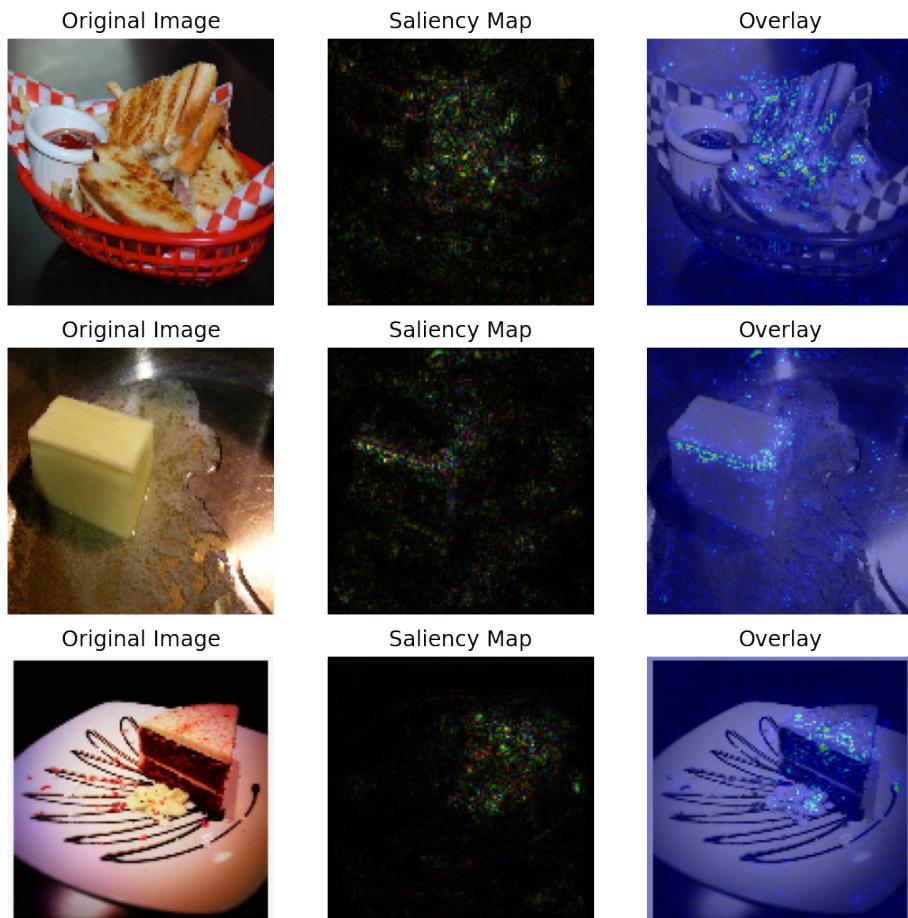
Machine Learning - Homework 5

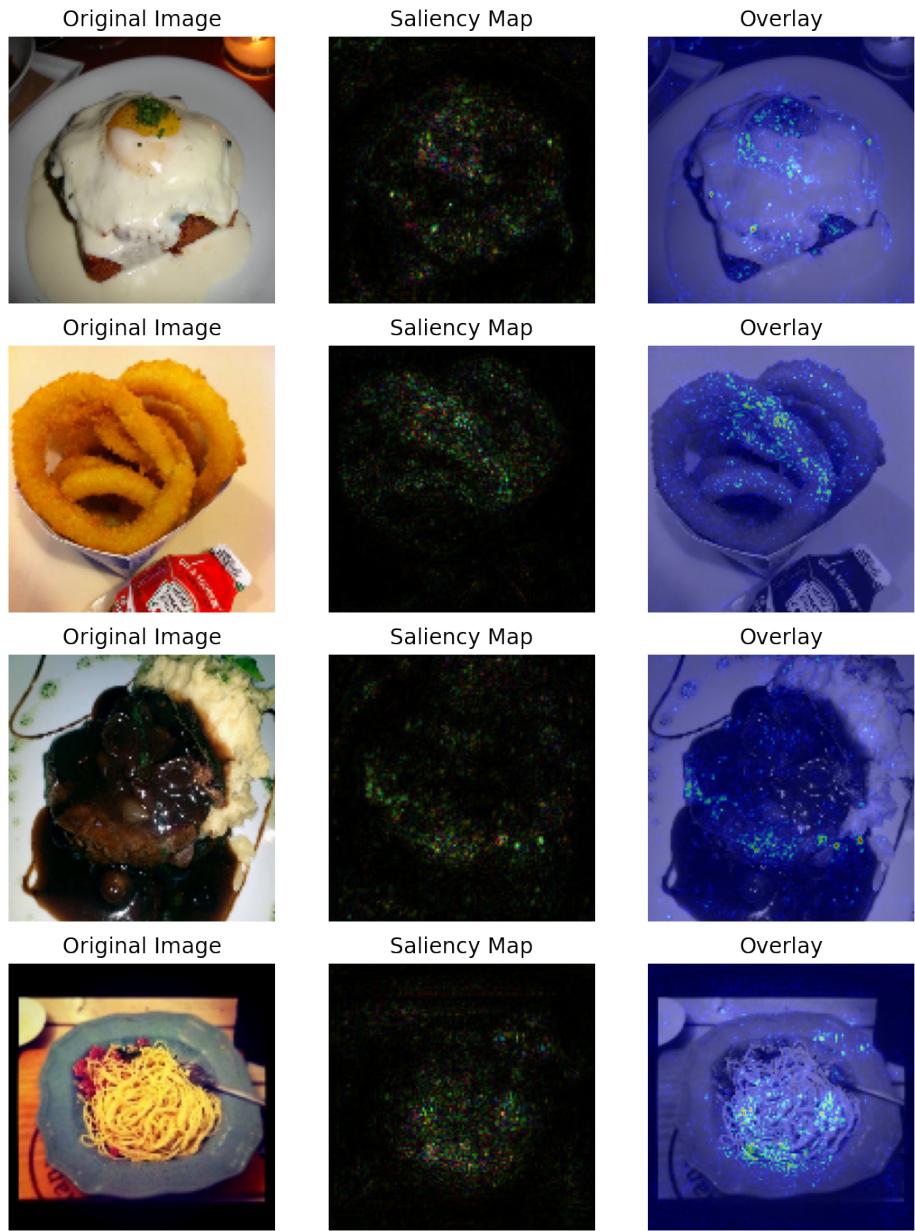
資工四 B05902023 李澤謙

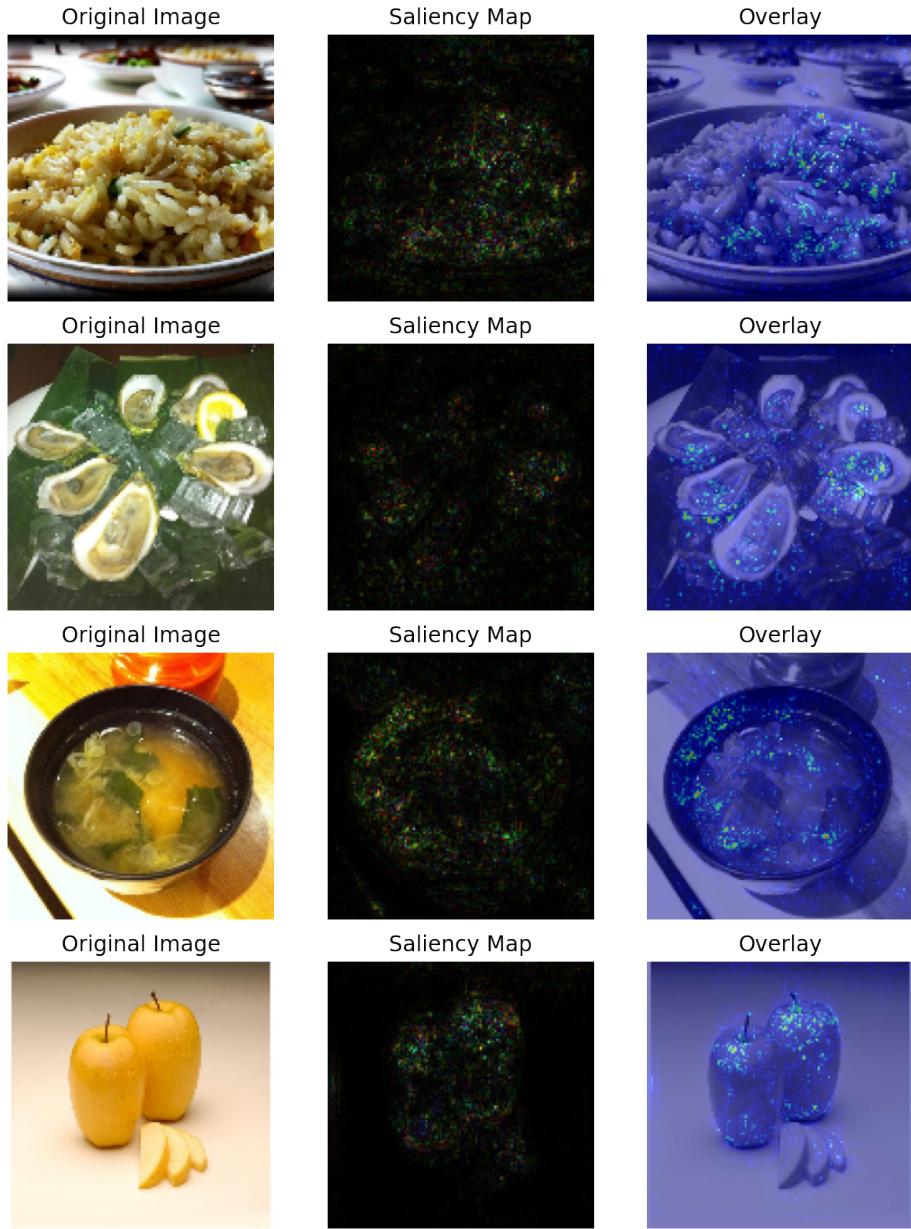
April 30, 2020

1. (2%) 從作業三可以發現，使用 CNN 的確有些好處，試繪出其 saliency maps，觀察模型在做 classification 時，是 focus 在圖片的哪些部份？

下圖為我從 11 個 class 之中各挑出一張照片所畫出的 saliency map：



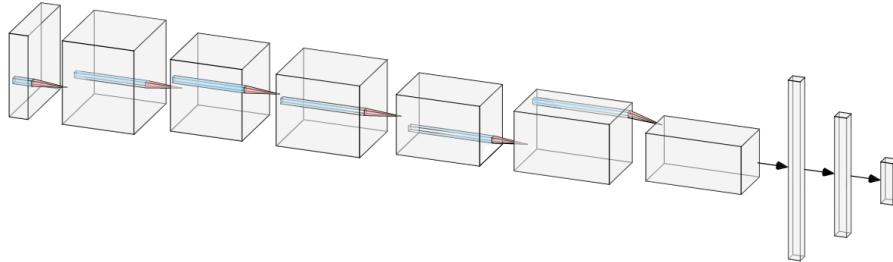




由上圖可以看出，CNN 確實有找出食物所在的位置，而非透過照片中的其它物品來進行 classification，並且，由上圖也可以看出，CNN 似乎容易專注於食物的輪廓上以此來進行 classification。

2. (3%) 承 (1)，利用上課所提到的 gradient ascent 方法，觀察特定層的 filter 最容易被哪種圖片 activate 與觀察 filter 的 output。

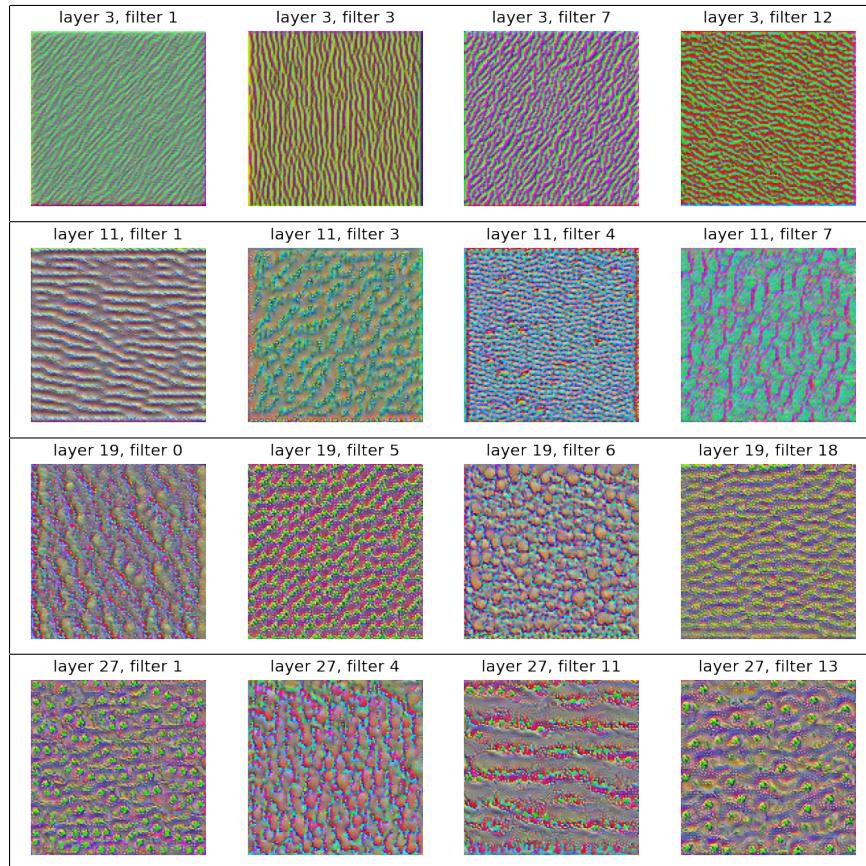
以下為我於 Homework 3 之中使用 PyTorch 所實作的 CNN 架構：



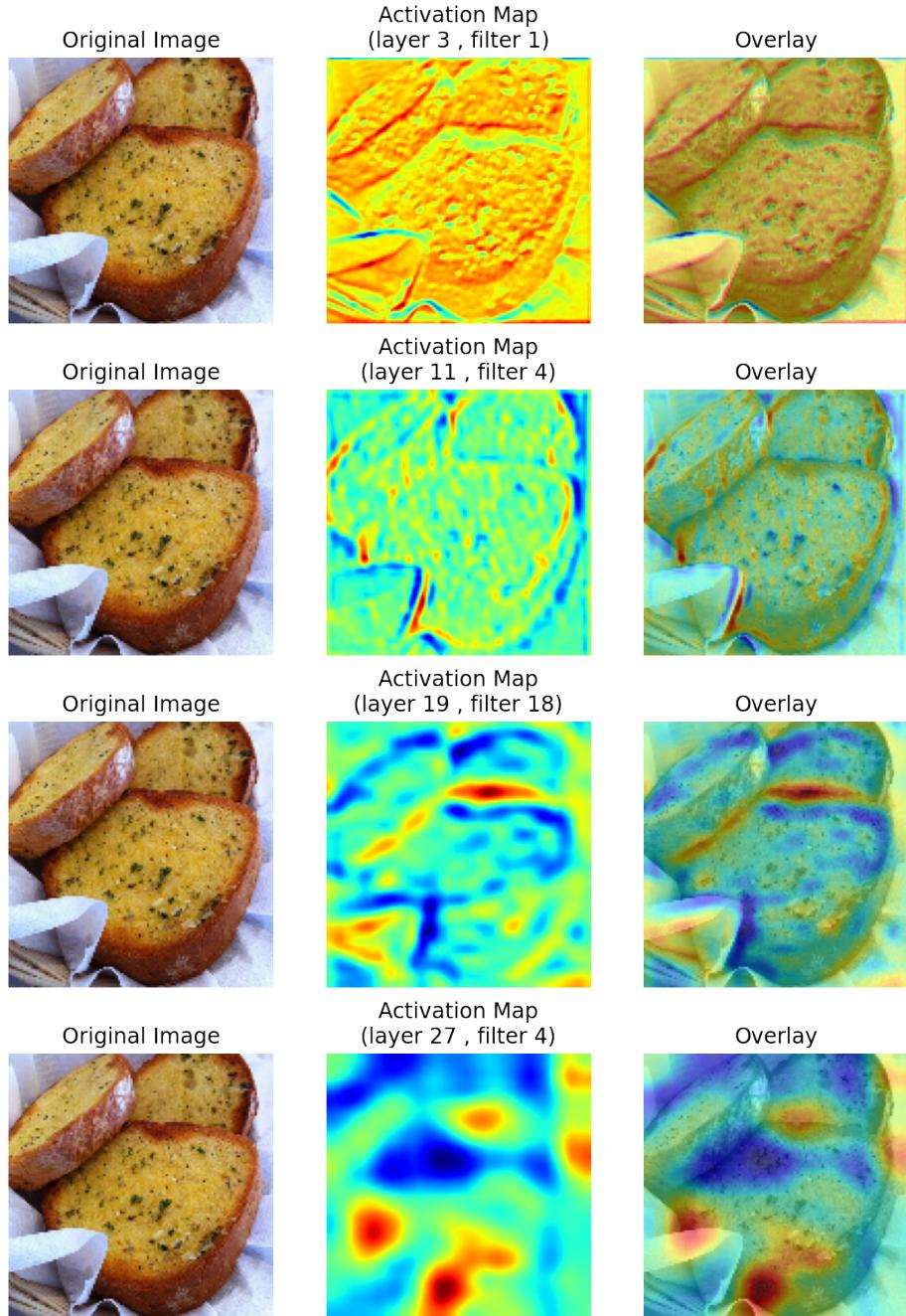
| |
|--|
| Conv2d(3 , 64 , kernel_size = 3 , stride = 1 , padding = 1) |
| BatchNorm2d(64) |
| ReLU() |
| Conv2d(64 , 64 , kernel_size = 3 , stride = 1 , padding = 1) |
| BatchNorm2d(64) |
| ReLU() |
| MaxPool2d(2) |
| Dropout2d(0.1) |
| Conv2d(64 , 128 , kernel_size = 3 , stride = 1 , padding = 1) |
| BatchNorm2d(128) |
| ReLU() |
| Conv2d(128 , 128 , kernel_size = 3 , stride = 1 , padding = 1) |
| BatchNorm2d(128) |
| ReLU() |
| MaxPool2d(2) |
| Dropout2d(0.1) |
| Conv2d(128 , 256 , kernel_size = 3 , stride = 1 , padding = 1) |
| BatchNorm2d(256) |
| ReLU() |
| Conv2d(256 , 256 , kernel_size = 3 , stride = 1 , padding = 1) |
| BatchNorm2d(256) |
| ReLU() |
| MaxPool2d(2) |
| Dropout2d(0.1) |
| Conv2d(256 , 512 , kernel_size = 3 , stride = 1 , padding = 1) |
| BatchNorm2d(512) |
| ReLU() |
| Conv2d(512 , 512 , kernel_size = 3 , stride = 1 , padding = 1) |
| BatchNorm2d(512) |
| ReLU() |
| MaxPool2d(2) |
| Dropout2d(0.5) |

| |
|------------------------------------|
| Linear(32768 , 1024 , bias = True) |
| BatchNorm1d(1024) |
| ReLU() |
| Dropout(0.5) |
| Linear(1024 , 11 , bias = True) |

我從以上的 CNN 中的第 3、11、19、27 (index 從 0 開始) 層 layer 中各挑出了 4 個 filter，將其所偵測的 pattern 畫出，其圖片如下：



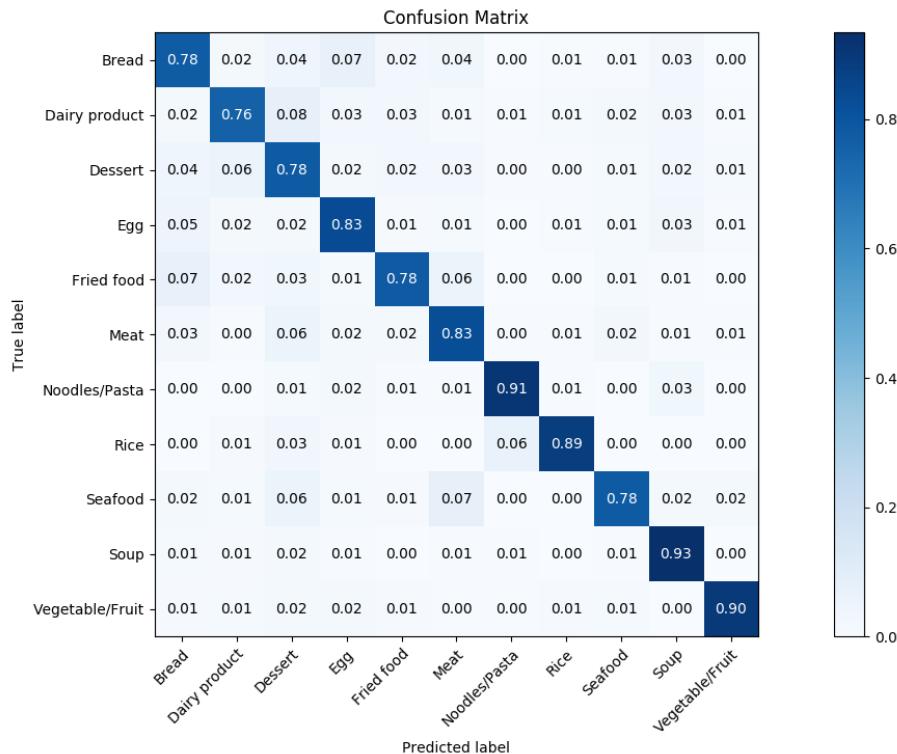
由上圖可以看出，不論是哪一層 layer，其中的 filter 所偵測的 pattern 大多皆為點或線條等簡單的紋路，其可能是因為 Homework 3 之中僅有 11 個 class，屬於比較簡單的 task，因此不需要特別複雜的 pattern 即可達到 classification，不過由上圖仍然可以看出，當 layer 的層數越大時，filter 所偵測的 pattern 其紋路似乎有越來越複雜的趨勢，且顏色也越來越複雜，因此可以推斷 CNN 之中，後面的 filter 會使用前面的 filter 所偵測到的簡單的 pattern 繼續組合為更複雜的 pattern，以此對照片達到 classification。接著，我從以上的 CNN 中的第 3、11、19、27 (index 從 0 開始) 層 layer 中各挑出了一個 filter，並將一張照片輸入 CNN 之中，觀察該張照片之中哪些位置可以使得以上各個 filter 的 activation 最大，其圖片如下：



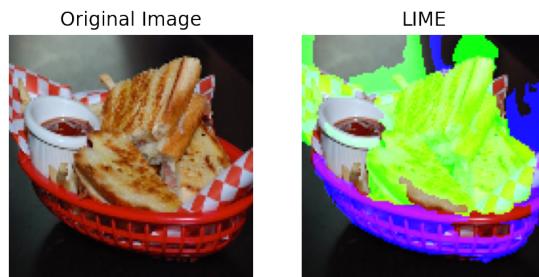
前段說到以上的 CNN 中的各個 filter 大多皆為偵測點或線條等簡單的紋路，並且後面的 filter 會使用前面的 filter 所偵測到的簡單的 pattern 繼續組合為更複雜的 pattern，故偵測的範圍也會更廣，以上的 activation map 可以應證這一點。

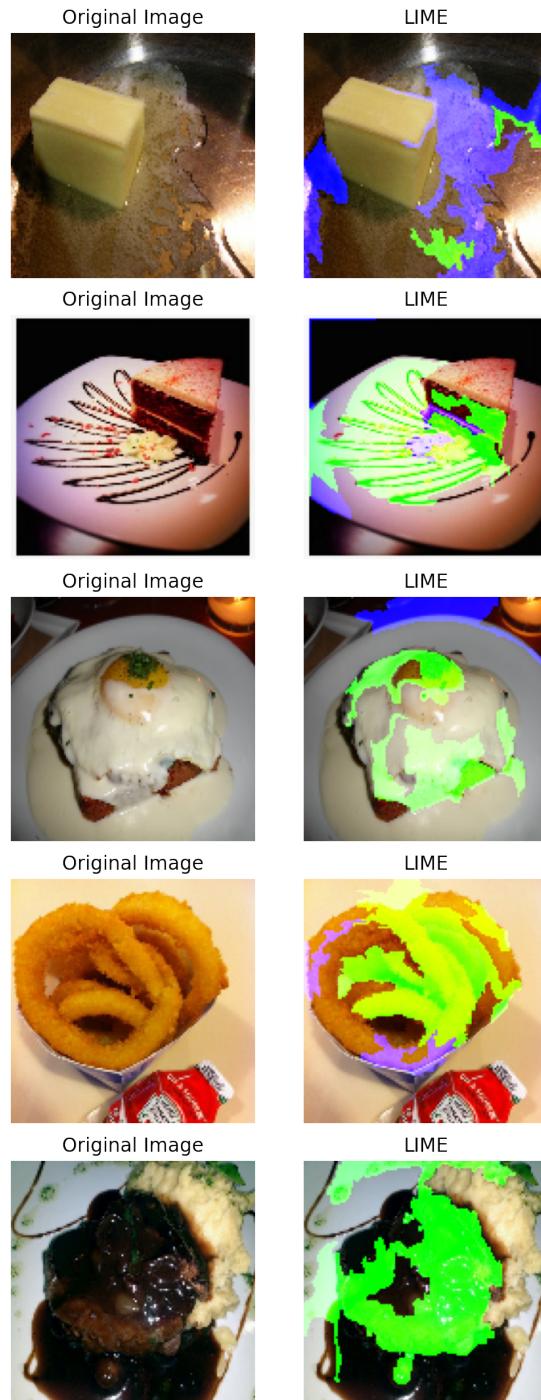
3. (2%) 請使用 Lime 套件分析你的模型對於各種食物的判斷方式，並解釋為何你的模型在某些 label 表現得特別好 (可以搭配作業三的 Confusion Matrix)。

以下為我於 Homework 3 之中所實作的 CNN (其架構如第 2 題所述) 其在 validation dataset 上所得到的 confusion matrix：



接著，下圖為我從 11 個 class 之中各挑出一張照片並使用 Lime 所畫出的圖片：







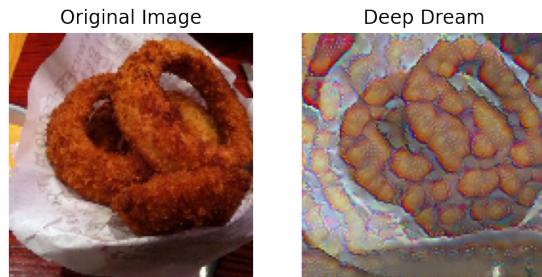
由上圖可以看出，CNN 確實有找出食物所在的位置並以此來進行 classification，

除此之外，其可能也會找出配菜或餐具等物品來幫助 classification。而由 confusion matrix 可以看出，CNN 在 noodles/pasta、rice、soup、vegetable/fruit 這 4 個 class 上的 performance 比較好，根據上圖，我猜測其原因為 CNN 在對這 4 個 class 的照片進行 classification 時，幾乎是完全專注這 4 種食物的主體而較少考慮到照片中的其它物品，並且這 4 種 class 的食物外觀變化不大，因此 CNN 可以在這 4 個 class 上得到不錯的 performance。

4. (3%)[自由發揮] 請同學自行搜尋或參考上課曾提及的內容，實作任一種方式來觀察 CNN 模型的訓練，並說明你的實作方法及呈現 visualization 的結果 (請附上 reference)。

我實作了簡化版的 deep dream，其實作方式如下：在我將照片輸入 CNN 之後，會將其中一層 convolutional layer 的 output 取出，並計算該 output 的 L2 norm，接著在固定 CNN 的 parameter 之下，進行 gradient ascent，即更新照片中 pixel 的值，使得上述的 L2 norm 數值變大，如此進行數個 iteration。其原理為：當我們想要觀察的 convolutional layer 其 output 的 L2 norm 變大時，該 convolutional layer 的 output 中為正的數值便會變得更大，而 output 中為負的數值則會變得更小，意即會使得該 convolutional layer 整體的 activation 變大，因此，在如此進行 gradient ascent 對照片進行更新之後，便能誇大化 CNN 在該張照片之中所見到的 pattern。在實作上有一些細節需要注意，例如不能對照片進行太多的更新 (例如可設定 convolutional layer 其 output 的 L2 norm 的上限)，以免會於結果中完全看不到原始照片的樣貌，除此之外，Google 所提出的 deep dream 的技術中，會將照片縮放為不同的大小與解析度，並調整 learning rate 等等，以加速 converge 的過程，此外，也可以控制 deep dream 中被誇大化的 pattern 的外觀 (<https://www.pytorchtutorial.com/deepdream-pytorch/> 之中有簡單的介紹)。下圖為我取了三張照片進行 deep dream 的結果：





於第 2 題之中我們發現 CNN 中的各個 filter 其所偵測的 pattern 大多皆為簡單的紋路，因此當我們對照片進行 deep dream 之後，照片中被誇大化的 pattern 大多皆為點、線條、圓圈等簡單的圖案。