Problem Set 4 Sample Report

Anonymous

April 19, 2021

1. (a) First, we assume that k is odd and that the vectors are zero-indexed. We will also use zero padding when the kernel extends over the input dimensions. We can define the equation for a specific element of the output x_{out_i} in the following way

$$x_{out_i} = \sum_{j=0}^{k-1} w_j x_s \mathbb{1}(0 \leq s < N)$$

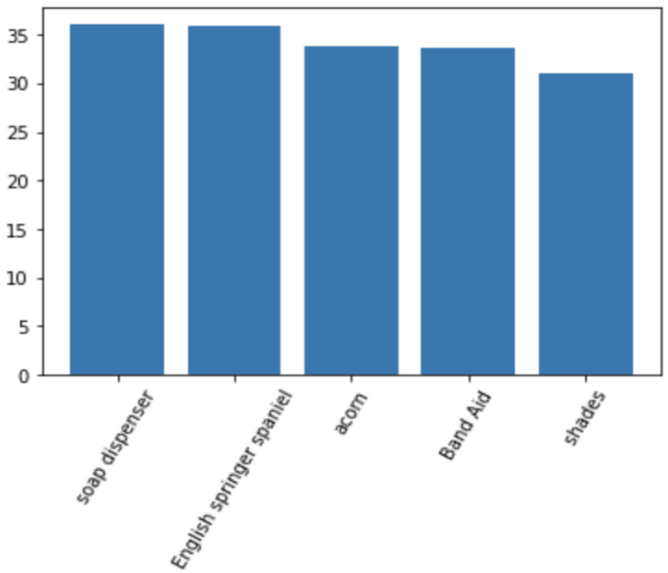
where $s = i - \frac{k-1}{2} + j$ and w_j is the j^{th} element of the flattened kernel W. This can alternatively be written as a linear operation $x_{out} = Mx_{in}$ where M is an $N \times N$ matrix of the form

$$\begin{bmatrix} w_{\frac{k-1}{2}} & w_{\frac{k+1}{2}} & \dots & 0 & \dots & & \\ & & & \dots & & & \\ 0 & \dots & w_0 & \dots & w_{k-1} & 0 & \dots \\ & & & \dots & & & \\ & \dots & 0 & \dots & w_{\frac{k-3}{2}} & w_{\frac{k-1}{2}} & \end{bmatrix}$$

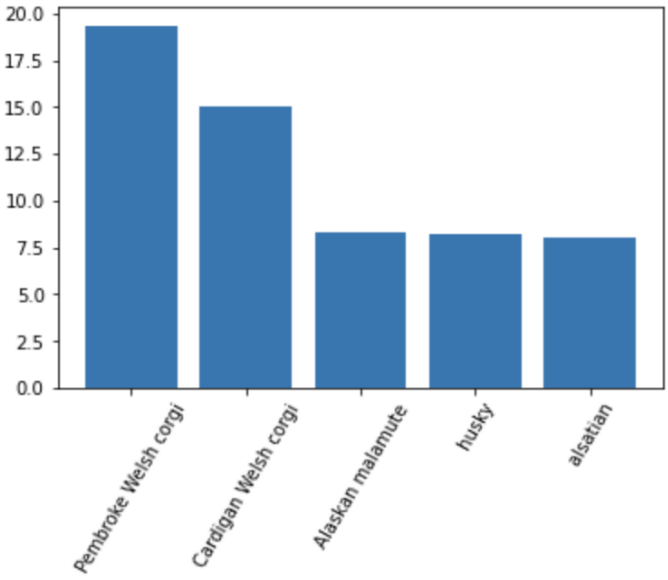
   (b) We will assume that the kernel of the max-pooling operation is of size $2k+1$ and that the vectors are zero-indexed but since x_{out} and x_{in} are different dimensions, we will not pad the input vector. Let $N(x_i) = \{x_{i-k}, ..., x_i, ...x_{i+k}\}$ be the neighborhood of a vector element. Then $x_{out_j} = \max N(x_{in_{j+k}})$ for all integers $0 \leq j < N - 2k - 1$

2. (a) The last layer has 2048 input features and 1000 outputs.

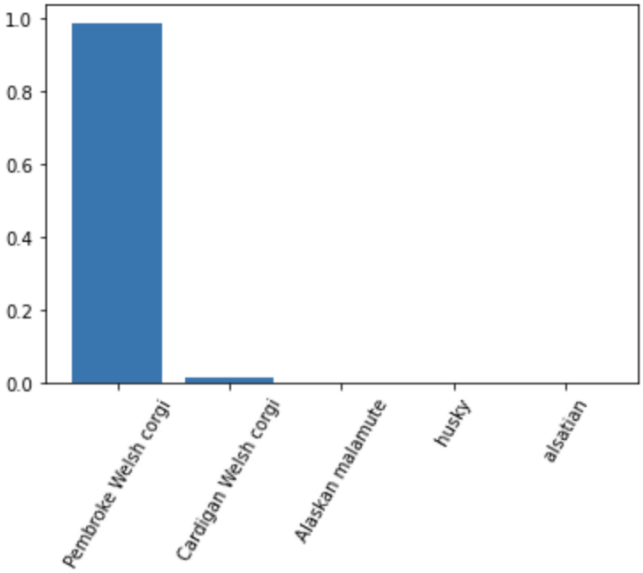
   (b) Below are the top predictions (before softmax) of my randomly instantiated model



(c) Below are the top predictions (before softmax) from the pretrained model.



Below are the top predictions after softmax. The probability of the top prediction (Pembroke Welsh Corgi) is 98.64%.



3. (a) Using the same assumptions as $1(a)$, we will find an expression for $\frac{\partial L}{\partial x_{in}} = \frac{\partial L}{\partial x_{out}} \frac{\partial x_{out}}{\partial x_{in}}$. Note that since this convolution operation can be represented as a linear operation $x_{out} = Mx_{in}$, This expression becomes $\frac{\partial L}{\partial x_{in}} = \frac{\partial L}{\partial x_{out}} M$ where M is defined as in $1(a)$.

   (b) Unfortunately, since the weights in the matrix M as we defined previously are shared between various inputs, calculating the gradient of the Loss with respect to the weights W is not as simple. Instead, we use the formulation $x_{out_i} = \sum_{j=0}^{k-1} w_j x_s \mathbb{1}(0 \leq s < N)$ where $s = i - \frac{k-1}{2} + j$ and w_j is the j^{th} element of the flattened kernel W. From this, we see that $\frac{\partial x_{out_i}}{w_j} = x_s \mathbb{1}(0 \leq s < N)$. That is, the derivative of a particular element of x_{out} with respect to weight w_j is $x_{i-\frac{k-1}{2}+j}$ if

$i - \frac{k-1}{2} + j$ is contained in the dimensions of the input and 0 otherwise. This gives that $\frac{\partial x_{out}}{W}$ is a $N \times 2k-1$ matrix A where the element at i, j is $x_s \mathbb{1}(0 \leq s < N)$. The result $\frac{\partial L}{\partial W} = \frac{\partial L}{\partial x_{out}} A$ is a $1 \times 2k-1$ vector where each element is the derivative of the loss with respect to an element of the kernel W. The weight update equation is $W^{i+1} = W^i + \eta(\frac{\partial L}{\partial x_{out}})^T$

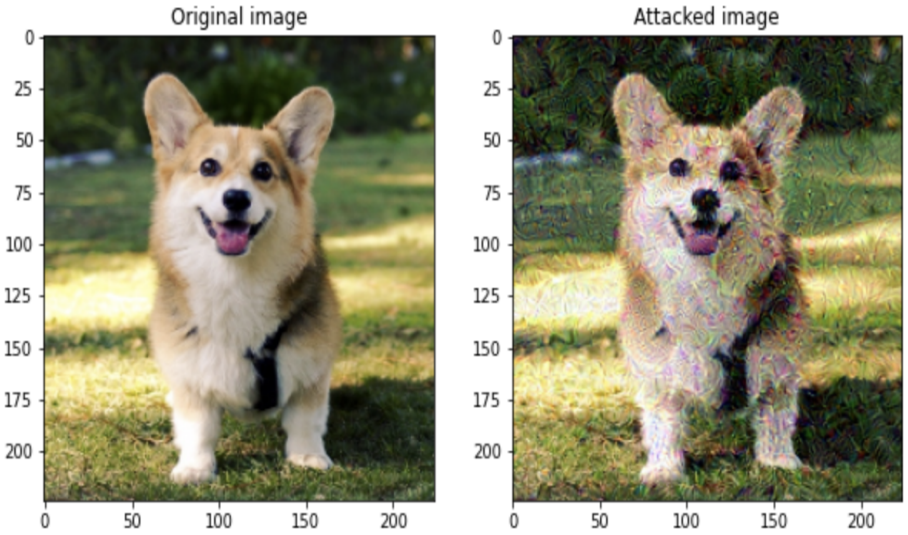
(c) I chose to use zero-padding so that each input element would produce an output element and therefore preserve the dimensions of the input.

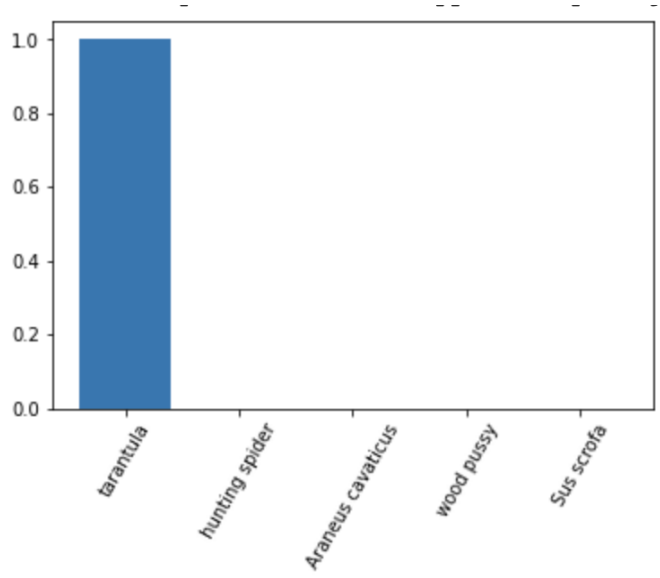
(d) To calculate $\frac{\partial L}{\partial x_{in}} = \frac{\partial L}{\partial x_{out}} \frac{\partial x_{out}}{\partial x_{in}}$ observe that $\frac{\partial x_{out_i}}{\partial x_{in_j}}$ is 1 if x_{in_j} is the maximum in the neighborhood of $x_{in_{i+k}}$ and 0 otherwise. Therefore, we can write $\frac{\partial x_{out_i}}{\partial x_{in_j}} = \mathbb{1}(j = \text{argmax } N(x_{in_{i+k}}))$ and $\frac{\partial x_{out}}{\partial x_{in}}$ is some matrix $N-2k \times N$ matrix B where $B_{i,j} = \mathbb{1}(j = \text{argmax } N(x_{in_{i+k}}))$ for $0 \leq j < N-2k-1$. So $\frac{\partial L}{\partial x_{in}} = \frac{\partial L}{\partial x_{out}} B$.

(e) Because the stride of the max-pooling was 1, I decided to not use any padding to fulfill the requirement that the input and ouput have different dimensions. Because their is no padding, the input has dimensions $1 \times N$ and the output has dimension $1 \times N - 2k$.

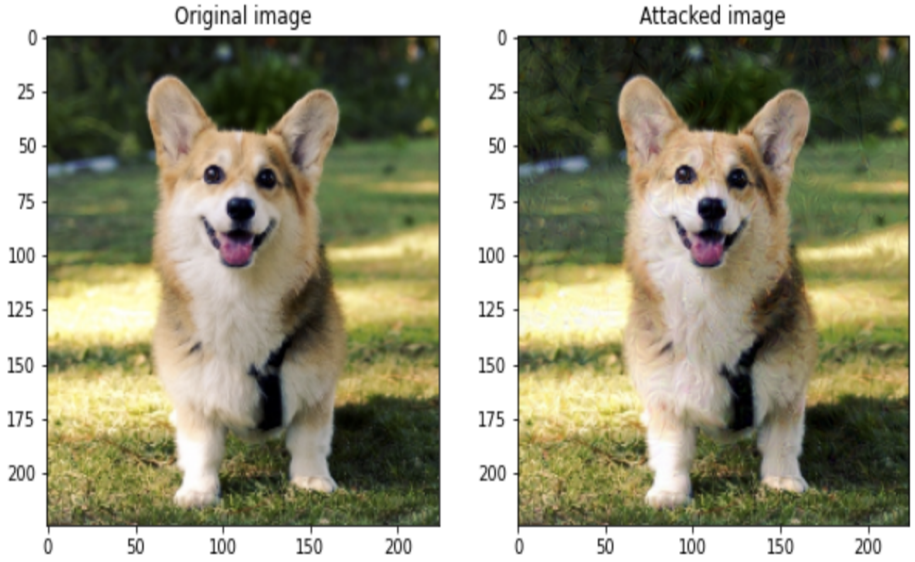
4. (a) Below is the original picture and the picture that is optimized to be classified as a tarantula along with the output predictions of the optimized image.

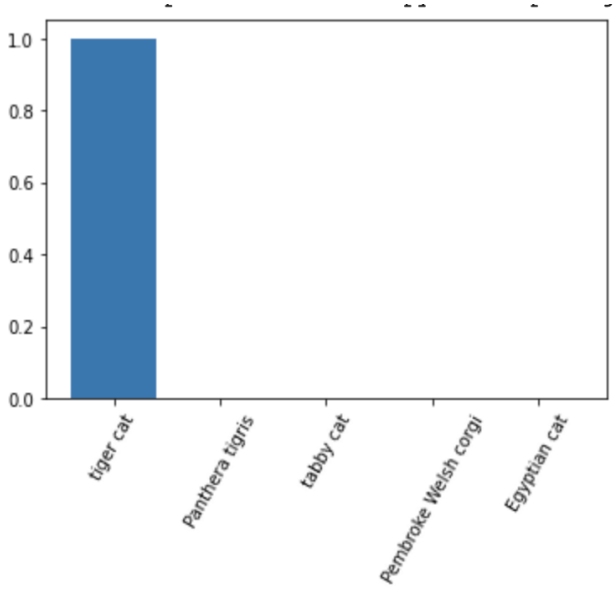
```
<matplotlib.image.AxesImage at 0x7fd408bacf10>
```





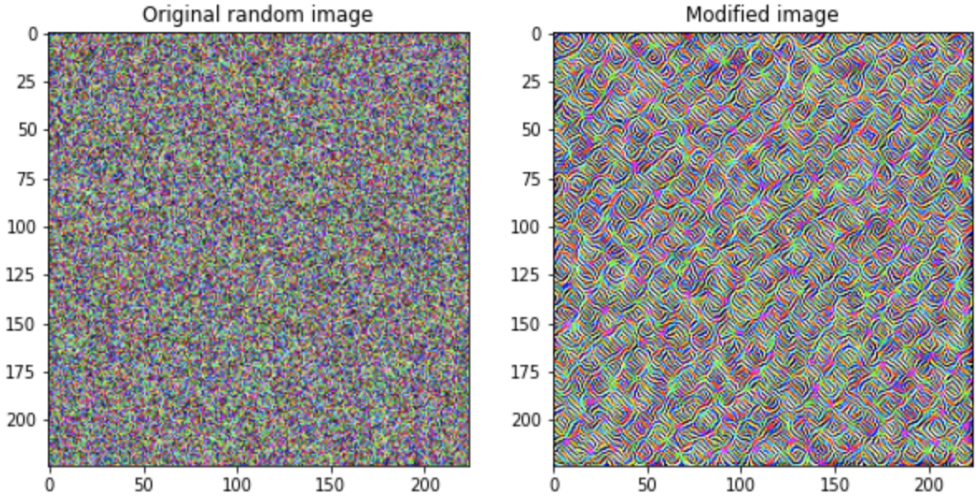
(b) Below is the original picture and the picture that is optimized to be classified as a tiger cat along with the output predictions of the optimized image.

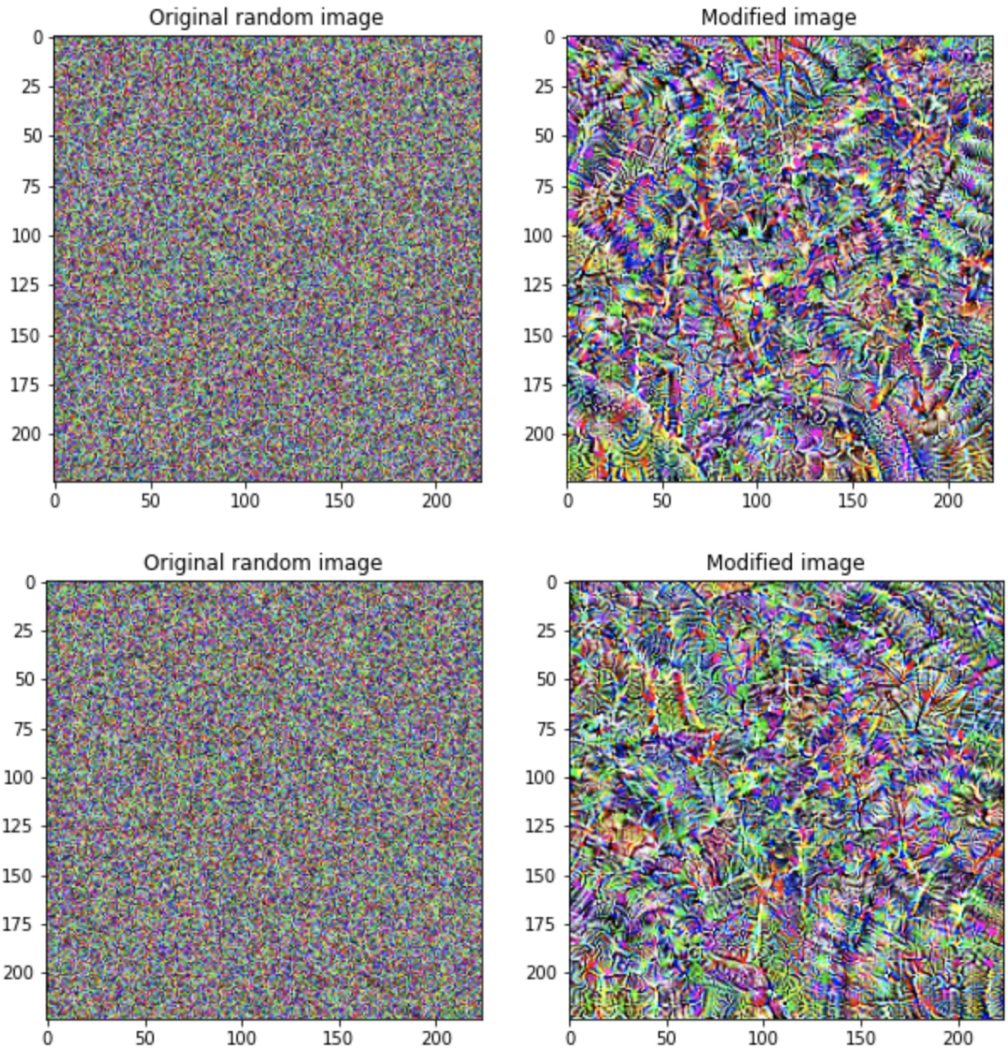




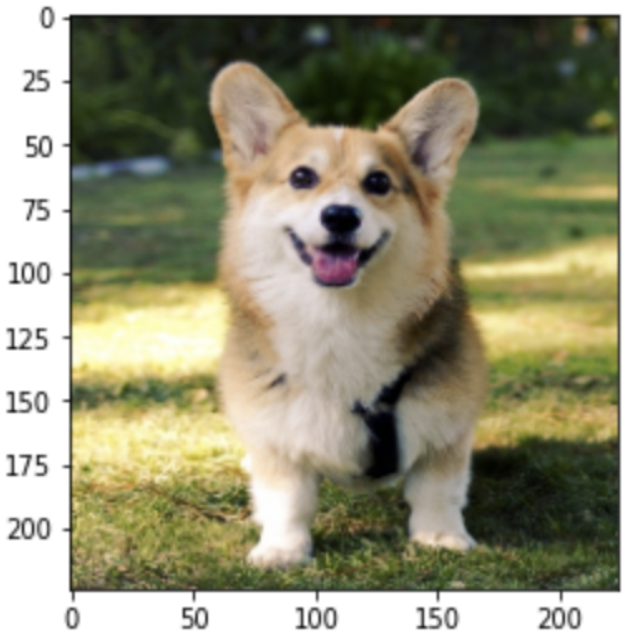
(c) The perturbed image from (b) looks more like the original input because the class of tiger cats is more similar to those of corgis. More specifically, they share many features (four legs, fur, etc.) and many parts of the model that are integral in identifying tiger cats are therefore also important in identifying corgis so the image will not need to be perturbed that much to activate those parts of the network. Tarantulas, on the other hand, are very different from corgis and so the image needs to be perturbed more significantly in order to activate parts of the network that are not activated much at all by images of corgis.

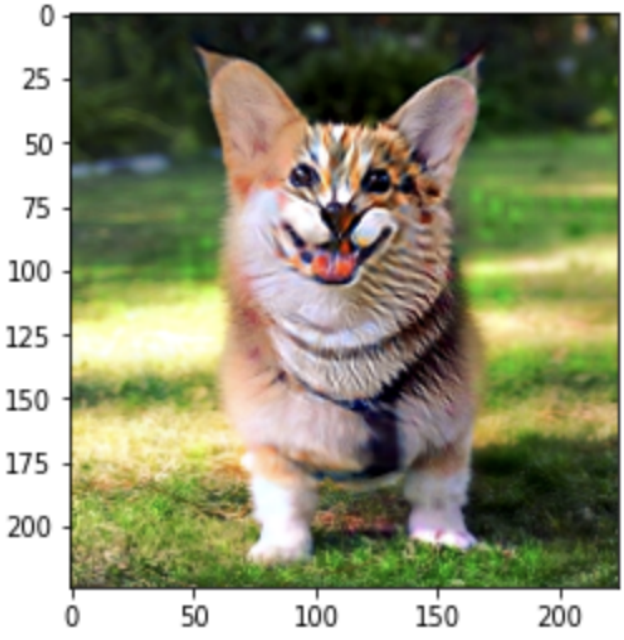
(d) Below are the results of the image when optimized to maximize the 0^{th}, 24^{th}, and 49^{th} layers of the network:

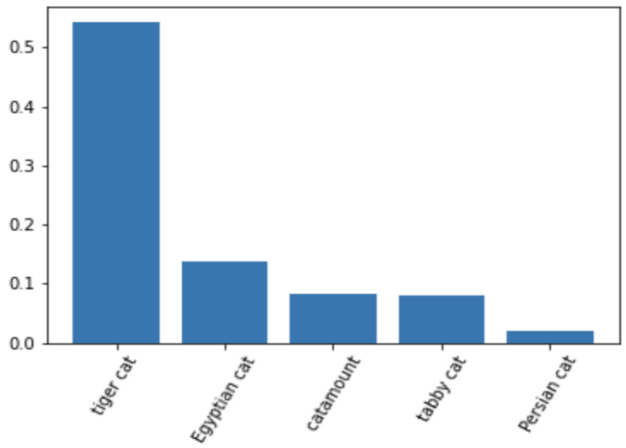




(e) Below is the result of perturbing the image to optimize for classification as tiger cat when using the robust network along with the distribution of the predictions.


0
25
50
75
100
125
150
175
200
0
50
100
150
200


0
25
50
75
100
125
150
175
200
0
50
100
150
200


0.5
0.4
0.3
0.2
0.1
0.0
tiger cat
Egyptian cat
catamount
tabby cat
Persian cat