Pset 3

Marlond Criollo ## Problem 1

Part A

The convolutional layer is the most popular module in computer
vision tasks. Assume your input x_{in} and output x_{out} are both 1D signals of size N, and
your kernel W has size k, where k is an odd number. Assume the stride size is 1. Find the equation for the forward propagation. You can omit the bias term of the convolutional layer.

We know that k is an odd number and that when the kernal passes over an input that index should be the middle of the kernel. The formula is as follows:

$$x_{out}[i] = \sum_{a=1}^{k} W[a] \cdot x_{in}\left[a - \frac{k-1}{2} - 1 + i\right]$$

$x_{in}[j] = 0$ when j<1 or j>k otherwise it gets $x_{in}[j]$

Problem 2

Part A

In the provided Python notebook, load the randomly initialized network and answer the following question: How many features are in the input of the last layer?

There are 2048 acting as inputs into the last layer

Part B

Run the Corgi image through the network. Include the generated plot in your report and answer the following question: What are the top-5 predictions?

The top 5 predictions are: Gong, dipper, armored combat vehicle, laptop computer, teddy bear.

Part C

Reload the network with pre-trained weights. These weights originate from a network trained on the ImageNet dataset. Run the Corgi image through the network. Then complete TODO1 and rerun the network. Include the two generated plots in your report (one displays the top-5 predicted logits, the other displays the top-5 predicted probabilities), and answer the following question: What are the top-5 predictions now? Please also include the code you completed in your report (i.e., the code for function output2prob).

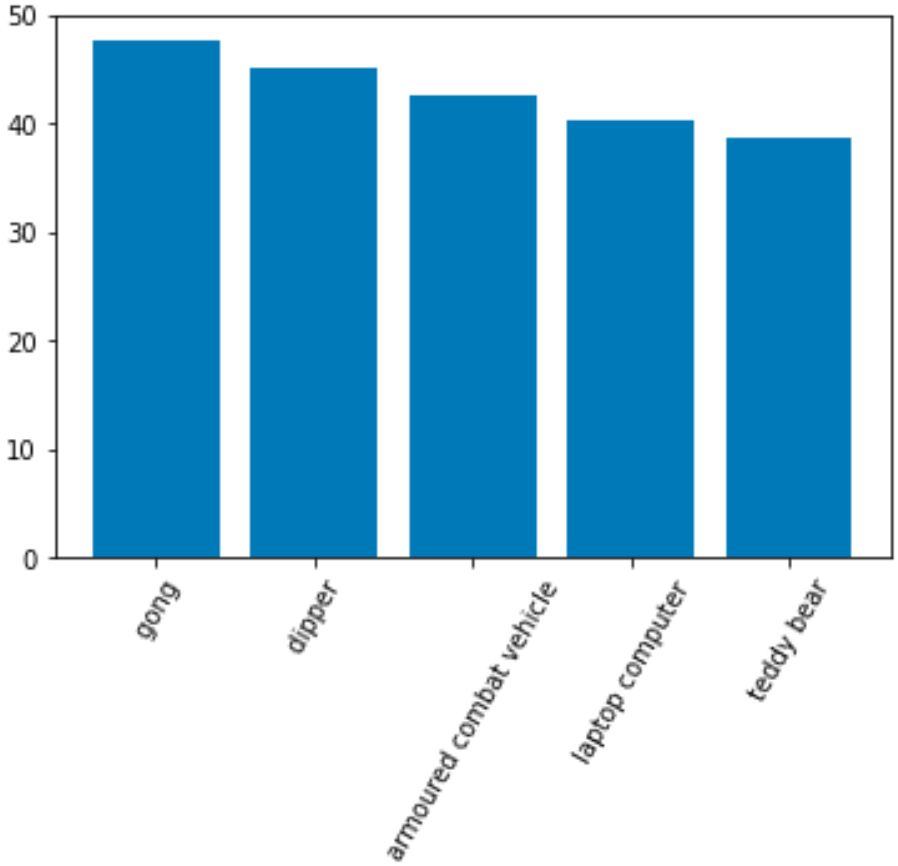


Figure 1: Top 5 predictions no training

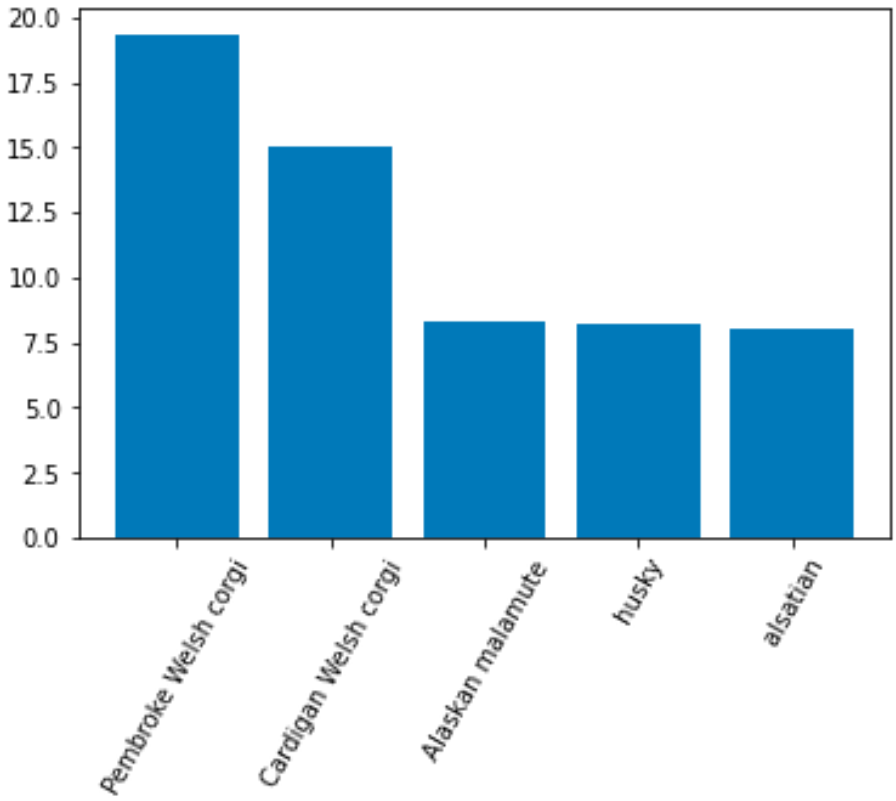


Figure 2: Logits

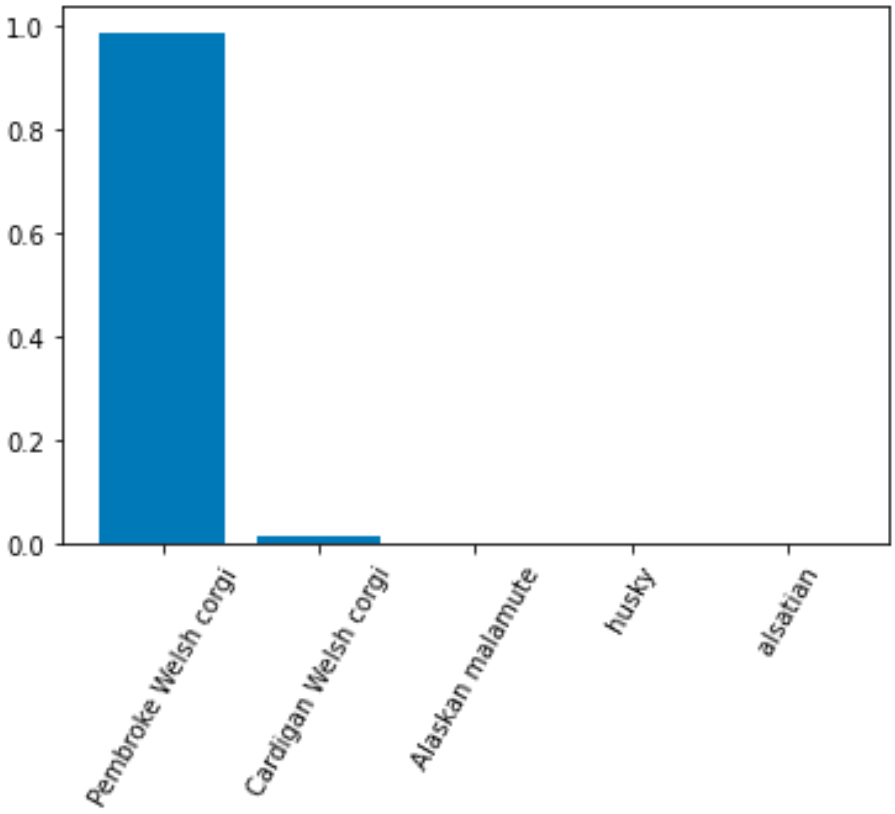


Figure 3: Probabilities

```
def output2prob(output):
### TODO1
# Your code here:
    prob = torch.nn.functional.softmax(output)
###
return prob
```

Top 5 now are: Pembroke Welsh Corgi, Cardigan Welsh corgi, alaskan malamute, husky, Alsatian

Problem 3

Part A

$$\frac{\partial C}{\partial x_{in}} = \frac{\partial C}{\partial x_{out}} \cdot \frac{\partial x_{out}}{\partial x_{in}}$$

$$= \frac{\partial C}{\partial x_{out}} \begin{bmatrix} \frac{\partial x_{out}[1]}{\partial x_{in}[1]} & \frac{\partial x_{out}[1]}{\partial x_{in}[2]} & \dots & \frac{\partial x_{out}[1]}{\partial x_{in}[N]} \\ \vdots & & & \vdots \\ \frac{\partial x_{out}[N]}{\partial x_{in}[1]} & \frac{\partial x_{out}[N]}{\partial x_{in}[2]} & \dots & \frac{\partial x_{out}[N]}{\partial x_{in}[N]} \end{bmatrix}$$

$$= \frac{\partial C}{\partial x_{out}} \begin{bmatrix} W\left[\frac{k-1}{2}+1\right] & W\left[\frac{k-1}{2}+2\right] & \dots & W\left[\frac{k-1}{2}+N\right] \\ \vdots & & & \vdots \\ W\left[\frac{k-1}{2}-(N-2)\right] & W\left[\frac{k-1}{2}-(N-1)\right] & \dots & W\left[\frac{k-1}{2}+1\right] \end{bmatrix}$$

Where $W[i] = 0$ outside of normal indexing and $W[i]$ otherwise

Part B

$$\frac{\partial C}{\partial W} = \frac{\partial C}{\partial x_{out}} \cdot \frac{\partial x_{out}}{\partial W_{ij}} = \frac{\partial C}{\partial x_{out}} \cdot x_{in}$$

$$= \frac{\partial C}{\partial x_{out}} \begin{bmatrix} \frac{\partial x_{out}[1]}{\partial W[1]} & \frac{\partial x_{out}[1]}{\partial W[2]} & \dots & \frac{\partial x_{out}[1]}{\partial W[k]} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial x_{out}[N]}{\partial W[1]} & \frac{\partial x_{out}[N]}{\partial W[2]} & \dots & \frac{\partial x_{out}[N]}{\partial W[k]} \end{bmatrix}$$

$$= \frac{\partial C}{\partial x_{out}} \begin{bmatrix} x_{in}\left[1-\frac{k-1}{2}\right] & x_{in}\left[2-\frac{k-1}{2}\right] & \dots & x_{in}\left[k-\frac{k-1}{2}\right] \\ \vdots & \vdots & \vdots & \vdots \\ x_{in}\left[N-\frac{k-1}{2}\right] & x_{in}\left[2-\frac{k-1}{2}\right] & \dots & x_{in}\left[k+N-\frac{k-1}{2}-1\right] \end{bmatrix}$$

This would be the gradients of the input and for the update rule of the

$$W^{k+1} \to W + \eta \left(\frac{\partial J}{\partial W}\right)^T$$

Where J is the sum of the losses associated with each training example

Part C

The boundaries can be handled like so, say we have a kernel $k = [333]$ to make this kernel into a matrix that can be applied to a vector x of length 6. The padding would be consistent of 0s and look like the following matrix, which can be padded even further for more dimensions of the input. I did this because otherwise the matrix would not be valid for tensor products.

$$\begin{bmatrix} 4 & 3 & 0 & 0 & 0 \\ 3 & 4 & 3 & 0 & 0 \\ 0 & 3 & 4 & 3 & 0 \\ 0 & 0 & 3 & 4 & 3 \\ 0 & 0 & 0 & 3 & 4 \end{bmatrix} \cdot \hat{X}$$

Problem 4

Part A

```
def generate_adversarial_example(model_fn, x, class_id, n_iter=200):
  """
  :param model_fn: a callable that takes an input tensor and returns the model logits.
  :param x: input tensor.
  :param class_id: the id of the target class.
  :return: a tensor for the adversarial example
  """
  for i in tqdm(range(n_iter)):
    ### TODO2
    # You should:
    # 1. Run input image/tensor x through the model
    # 2. Define the loss or the objective you want to maximize
    # 3. Compute the gradient of the objective with respect to x using
    #    torch.autograd.grad
    #
    # Your code here:
    logit = model_fn(x)
    loss = torch.nn.functional.softmax(logit, dim=1)[:,class_id]
    gradient, = torch.autograd.grad(-loss,x)
    ###
    x = step.step(x, -1*gradient)
  return x
```

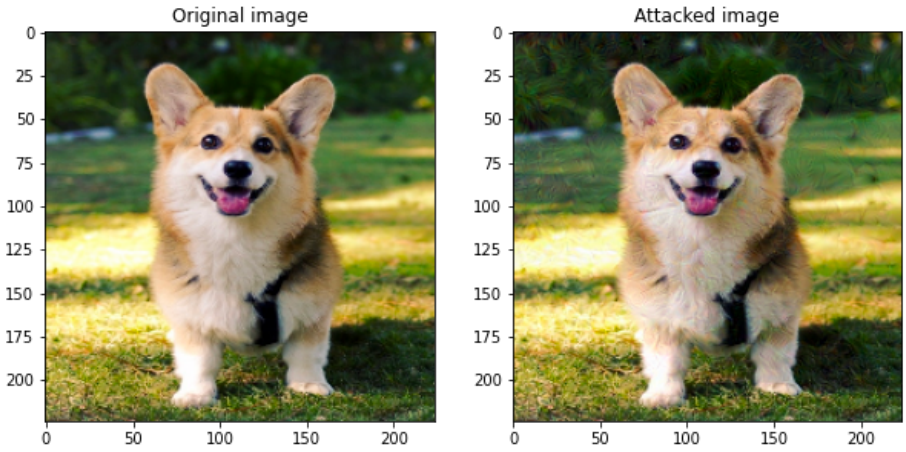


Figure 4: Images

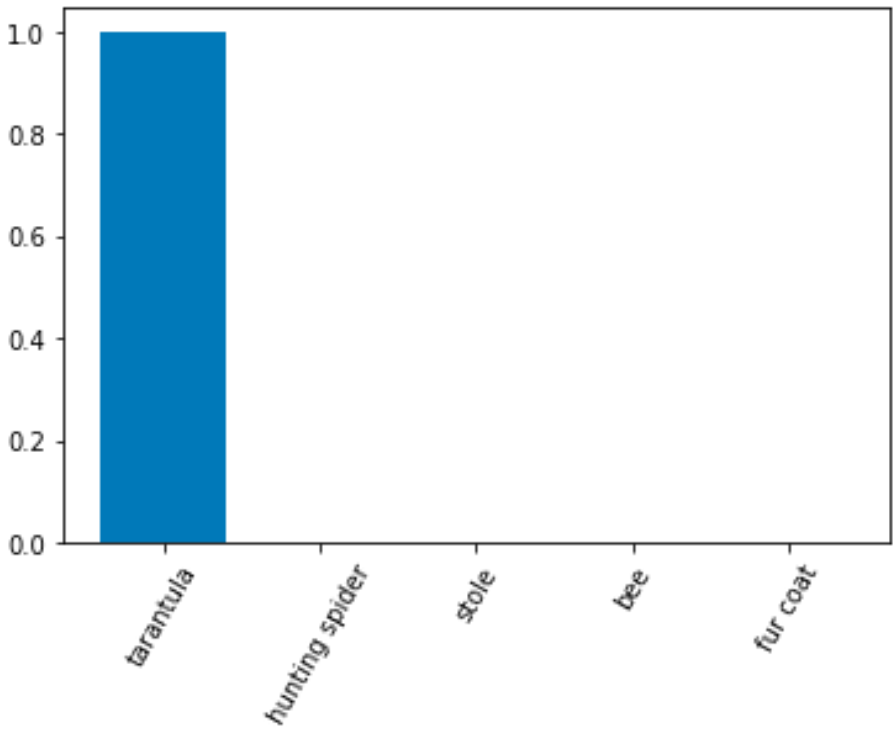


Figure 5: Probabilities

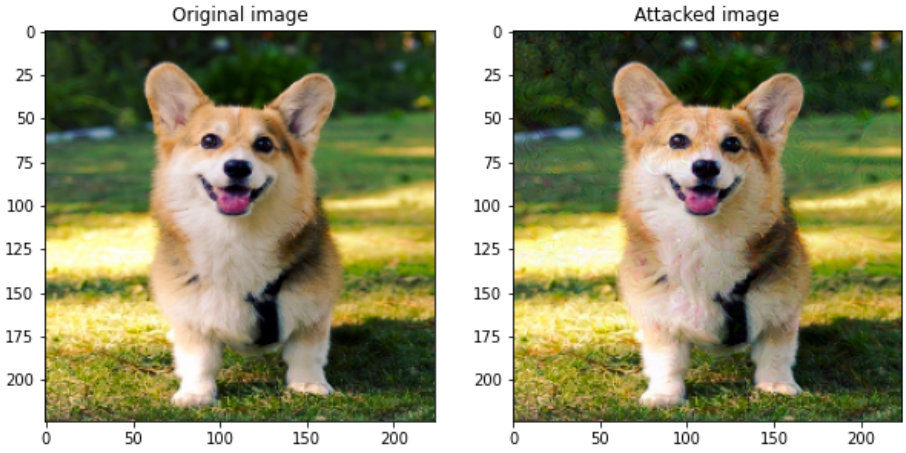
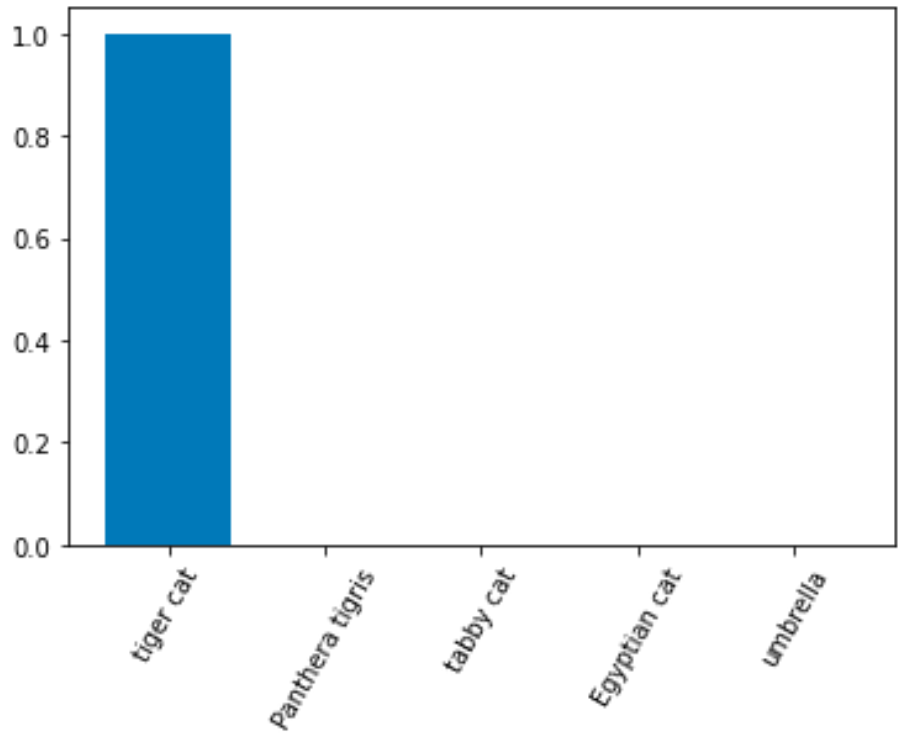


Figure 6: image

Part B



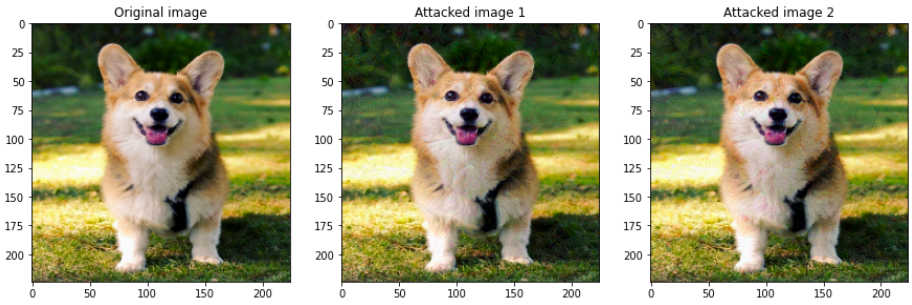


Figure 7: Compared Images

Part C

I have no way of telling the images apart from each other, which can create huge issues when these training databases are comprised of very large pools of images which are sometimes pulled straight from the internet. When using this kind of technology in a security setting we make our selves susceptible to incorrect classifications and holes within our security. When presented with an image like the above we would not be able to tell anything was wrong with it until runtime which is never a good thing.