Problem Set 5

Basil R. Yap
50.021 Artificial Intelligence - Term 8

June 19, 2018

1 Theory Component

[**Q1**]. Consider the following 3×3 filter.

$$w = \begin{bmatrix} -3 & 0 & 3 \\ -4 & 0 & 4 \\ -3 & 0 & 3 \end{bmatrix}$$

This filter w is applied to a grayscale image shown in Figure 1. Assume that the dimension the image in Figure 1 is way larger than 3×3. We can express the image in terms of matrix M, each element is numbered between 0 to 1 (0 being completely black and 1 being completely white). We are applying convolution of the filter w to M : $(M * w)$. Answer the following questions,

1. For which part of the image will the filter return a number thats furthest possible from zero (very positive or very negative)? (ignore the arrow and the words, thats for the next question) Give a max of 3 sentences explanation.
2. Will the convolution output at the location indicated in Figure 1 be positive, negative, or zero in value? Give a max of 3 sentences explanation.

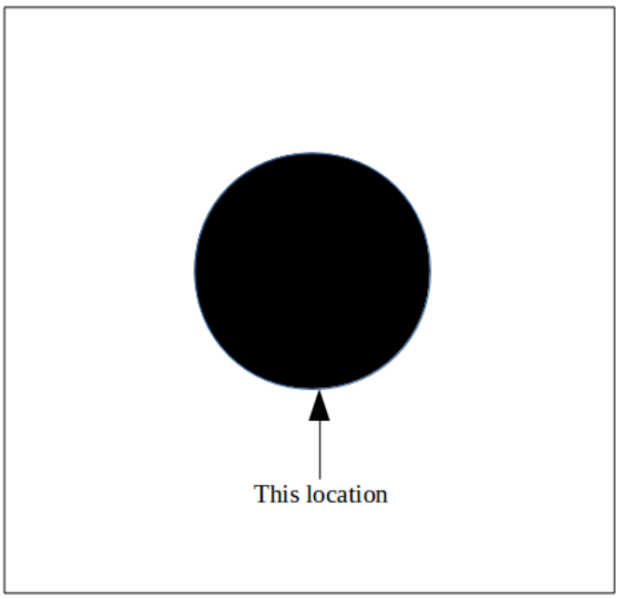

Figure 1: Figure for Q1

Solution

1. The parts with the largest magnitude from zero lies at the rightmost and leftmost parts of the circle. The rightmost having the most negative value, while the leftmost has the most positive value.

2. That location will have zero in value, as the values at that point would have values which are symmetrical along the columns of the filter. The filter, having equal negative and positive multiplier, will result in a zero sum when convoluted.

[Q2]. Figure 2 illustrates the before and after effect of a blurring filter when applied on a greyscale image. Blurring is an operation that makes strong edges weaker by some kind of averaging. Black pixels take a value of 0 and white pixes take a value of 1. Assume that the blurring effect is done using the following 5 by 5 filters with stride 1, and there's enough zero padding on the image before processing so that the filter will fit on the edges.
Determine whether each of the filters below can or cannot give a blurring effect and give your reason in not more than 2 sentences :

1. $\frac{1}{10}\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & -4 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$

2. $\frac{1}{4}\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$

3. $\begin{bmatrix} 0 & 5 & 0 & 5 & 0 \\ 0 & 2 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & -2 & 0 & -2 & 0 \\ 0 & -5 & 0 & -5 & 0 \end{bmatrix}$

4. $\frac{1}{150}\begin{bmatrix} 3 & 4 & 8 & 4 & 3 \\ 4 & 7 & 9 & 7 & 4 \\ 8 & 9 & 10 & 9 & 8 \\ 4 & 7 & 9 & 7 & 4 \\ 3 & 4 & 8 & 4 & 3 \end{bmatrix}$

before filter

after filter

Solution

1. Cannot result in blurring effect. This filter takes a huge weight for the central pixel in relation to surrounding pixels, preserving the overall contrast of the image but inverting its colour. This filter may result in sharper contrast than the original image.

2. Can result in blurring effect. This filter averages the value of the central pixel indiscriminately with other pixels from the bottom right region. Blurring will occur toward the bottom and right side of the image.

3. Cannot result in blurring effect. Given a vertical edge, the filter has a high likelihood of returning zero or close to zero in value. The low value regions will create huge contrast and not contribute to a blurring effect. This filter is used to detect horizontal edges while downsampling the resolution of the image.

4. Can result in blurring effect. This filter averages the value of the central pixel with all other pixels in the filter, but with emphasis on the central pixel and the pixels surrounding it.

[**Q3**] Does the following neural networks suffer strongly from the vanishing gradient problem? Give your reason in not more than 2 sentences.

1. 1-Layer Feed-Foward NN
2. Very Deep Feed-Forward NN
3. Recurrent NN
4. LSTM NN
5. ResNet

Solution

1. Does not suffer greatly from the vanishing gradient problem. Backpropagation only occurs back one layer.

2. Suffers greatly from vanishing gradient problem. With a significant number of layers, the changes to the gradient of earlier layers are greatly diminished.

3. Suffers greatly from vanishing gradient problem. Due to the time component of the RNN, backpropagation through time has more parameters to consider at each step and results in the gradients of earlier layers to change less.

4. Does not suffer greatly from the vanishing gradient problem. Due to the presence of the forget gate in LSTM, the change in gradient remains constant in recurrence and will not vanish as it propagates back to the earlier layers.

5. Does not suffer greatly from the vanishing gradient problem. With the identity shortcut connection, the gradients of earlier layers would be less likely to be heavily diminished during backpropagation through many layers.

2 Coding Component

```
'''
00 - The Fool
Fooling classifier code
for homework 5
'''
import torch
import PIL.Image as Image
import PIL.ImageChops as ImChop

import torchvision
from torchvision import models, transforms, utils

import numpy as np
import matplotlib.pyplot as plt
import getimagenetclasses as ginc

model_transform = transforms.Compose([
    transforms.Resize(224),
    transforms.CenterCrop(224),
    transforms.ToTensor(),
    transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225]),
])

inv_transform = transforms.Compose([transforms.Normalize(mean=[0., 0., 0.],
                                                         std=[1 / 0.229, 1 /
                                    transforms.Normalize(mean=[-0.485, -0.456
                                                         std=[1., 1., 1.]),
                                    transforms.ToPILImage(),
                                    ])

resize_transform = transforms.Compose([
    transforms.Resize(224),
    transforms.CenterCrop(224)
])

i2s, s2i, s2d = ginc.parsesynsetwords('synset_words.txt')


def get_descript(class_no):
    s = i2s[class_no]
    desc = s2d[s]
    return desc
```

```
def get_trained_resnet(use_gpu=True):
    model_ft = models.resnet18(pretrained=True)
    if use_gpu:
        model_ft = model_ft.try_cuda()
    return model_ft


def plot_with_difference(original, fooled):
    '''
    Takes the normalised tensor of two images and plot them
    '''
    fooled_v = fooled.clone().view(3, 224, 224).cpu().detach()
    fooled_back = inv_transform(fooled_v)
    original_back = inv_transform(original.clone())

    plt.figure()
    plt.subplot(131)
    plt.imshow(original_back)
    plt.title('original')

    boi_diff = ImChop.difference(fooled_back, original_back)
    plt.subplot(132)
    plt.imshow(boi_diff)
    plt.title('difference')

    plt.subplot(133)
    plt.imshow(fooled_back)
    plt.title('fooling')
    print("Total value difference:", np.array(boi_diff).sum(),
          "\nAverage value difference:", np.array(boi_diff).mean())


# add function to torch.Tensor class
# python gets stuck after getting the second cuda error
# so after the first error, this function should always
# return self
if __name__ == "__main__":
    have_cuda = True

    def try_cuda(self):
        global have_cuda
        if have_cuda:
            try:
```

```
            with_cuda = self.cuda()
            return with_cuda
        except Exception as e:
            have_cuda = False
            print("try_cuda failed:", e, "\n proceeding without cuda")
    return self

torch.Tensor.try_cuda = try_cuda
torch.nn.Module.try_cuda = try_cuda
```