# Problem Set 3

Basil R. Yap
50.021 Artificial Intelligence - Term 8

May 27, 2018

## 1 Theory Component

**[Q1]**. Consider the following CNN that has:

1. Input of $14 \times 14$, with 30 channels.
2. A convolutional layer $C$ with 12 filters, each of size $4 \times 4$. The convolution zero-padding is 1 and the stride is 2, followed by a relu activation.
3. A max pooling layer $P$ that is applied over each of the $C$'s output feature maps, using $3 \times 3$ receptive fields and stride 2.

a) What is the total size of $C$'s output feature map?

b) What is the total size of $P$'s output feature map?

Now we want to compute the overhead of the above CNN in terms of floating point operation (FLOP). FLOP can be used to measure computer's performance. A decent processor nowadays can perform in Giga-FLOPS, that means billions of FLOP per second. Assume the inputs are all scalars (we have $14 \times 14 \times 30$ scalars as input), we have the computational cost of:

1. 1 FLOP for a single scalar multiplication $x_i \cdot x_j$
2. 1 FLOP for a single scalar addition $x_i + x_j$
3. $(n-1)$ FLOPs for a max operation over $n$ items: $\max\{x_1, ..., x_n\}$

c) How many FLOPs layer $C$ and $P$ cost in total to do one forward pass?

**Solution**

a) Output size: $O = \left\lfloor \frac{(W-K+2P)}{S} \right\rfloor + 1$

$O:$ Output size
$W:$ Input size
$K:$ Filter size
$P:$ Padding
$S:$ Stride

$$\begin{aligned} O &= \left\lfloor \frac{14-4+2}{2} \right\rfloor + 1 \\ &= 7 \end{aligned}$$

Size of $C$'s feature map: $7 \times 7 \times 12 = 588$

b) Output size: $O = \left\lfloor \frac{W-K}{S} \right\rfloor + 1$

$O:$ Output size
$W:$ Input size
$K:$ Filter size
$S:$ Stride

$$\begin{aligned} O &= \left\lfloor \frac{7-3}{2} \right\rfloor + 1 \\ &= 3 \end{aligned}$$

Size of $C$'s feature map: $3 \times 3 \times 12 = 108$

c) One convolution filter operation:

$$\begin{aligned} CF &= \text{Filter size} + (\text{Filter size} - 1) \\ &= 16 + 15 \\ &= 31 \end{aligned}$$

One convolution operation:

$$\begin{aligned} C &= CF \times \text{Output size} \\ &= 31 \times 49 \\ &= 1519 \end{aligned}$$

One ReLU operation:

$$\begin{aligned} R &= \text{Input size} \\ &= 7 \times 7 \\ &= 49 \end{aligned}$$

One max pooling operation:

$$\begin{aligned} MP &= (\text{Filter size} - 1) \times \text{Output size} \\ &= 8 \times 9 \\ &= 72 \end{aligned}$$

Total FLOPs:

$$\begin{aligned} \text{Total} &= (C + R) \times \text{Number of filters} \times \text{Channels} + MP \times \text{Number of filters} \\ &= (1519 + 49) \times 12 \times 30 + 72 \times 12 \\ &= 565344 \end{aligned}$$

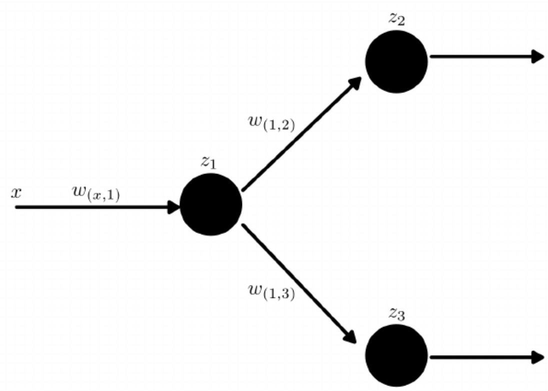


Figure 1: Mini Neural Network

**[Q2]**. Refer to the neural network at figure 1 with input $x \in \mathbb{R}^1$. The activation function for $z_1, z_2$, and $z_3$ is the sigmoid function: $\frac{1}{1+e^{-w \cdot x}}$,

$$h(x) = \frac{1}{1+e^{-x}} \tag{1}$$
$$z_1 = h(x \cdot w_{(x,1)}) \tag{2}$$
$$z_2 = h(z_1 \cdot w_{(1,2)}) \tag{3}$$
$$z_3 = h(z_1 \cdot w_{(1,3)}) \tag{4}$$

For the error $E$, instead of using the softmax function we learned in class, we use the quadratic error function for regression purpose,

$$E = \sum_{i \in data} ((z_2 - y_{2i})^2 + (z_3 - y_{3i})^2)$$

**[6p]** Write down an expression for the gradients of all three weights: $\frac{\partial E}{\partial w_{(x,1)}}, \frac{\partial E}{\partial w_{(1,2)}}, \frac{\partial E}{\partial w_{(1,3)}}$.

## Solution

$$\begin{aligned}
h(x) &= \frac{1}{1+e^{-x}} \\
h'(x) &= \frac{\delta h(x)}{\delta x} \\
&= \frac{e^{-x}}{(1+e^{-x})^2} \\
&= \frac{(1+e^{-x}) - 1}{(1+e^{-x})^2} \\
&= \frac{1}{1+e^{-x}} - \frac{1}{(1+e^{-x})^2} \\
&= h(x) - h(x)^2
\end{aligned}$$

$$\frac{\delta z_1}{\delta w_{(x,1)}} = xh'(x \cdot w_{(x,1)})$$

The value of $z_1$ does not change with respect to $w_{(1,2)}$ and $w_{(1,3)}$,

$$\therefore \frac{\delta z_2}{\delta w_{(1,2)}} = z_1 h'(z_1 \cdot w_{(1,2)})$$

$$\therefore \frac{\delta z_3}{\delta w_{(1,3)}} = z_1 h'(z_1 \cdot w_{(1,3)})$$

$$\begin{aligned}
E &= \sum_{i \in data} ((z_2 - y_{2i})^2 + (z_3 - y_{3i})^2) \\
\frac{\delta E}{\delta z_2} &= \sum_{i \in data} 2(z_2 - y_{2i}) \\
\frac{\delta E}{\delta z_3} &= \sum_{i \in data} 2(z_3 - y_{3i}) \\
\frac{\delta z_2}{\delta z_1} &= w_{(1,2)} h'(z_1 \cdot w_{(1,2)}) \\
\frac{\delta z_3}{\delta z_1} &= w_{(1,3)} h'(z_1 \cdot w_{(1,3)})
\end{aligned}$$

$$\begin{aligned}
\frac{\delta E}{\delta w_{(1,2)}} &= \frac{\delta E}{\delta z_2} \frac{\delta z_2}{\delta w_{(1,2)}} \\
&= z_1 h'(z_1 \cdot w_{(1,2)}) \sum_{i \in data} 2(z_2 - y_{2i}) \\
&= z_1 (h(z_1 \cdot w_{(1,2)}) - h(z_1 \cdot w_{(1,2)})^2) \sum_{i \in data} 2(z_2 - y_{2i})
\end{aligned}$$

$$\begin{aligned}
\frac{\delta E}{\delta w_{(1,3)}} &= \frac{\delta E}{\delta z_3} \frac{\delta z_3}{\delta w_{(1,3)}} \\
&= z_1 h'(z_1 \cdot w_{(1,3)}) \sum_{i \in data} 2(z_3 - y_{3i}) \\
&= z_1 (h(z_1 \cdot w_{(1,3)}) - h(z_1 \cdot w_{(1,3)})^2) \sum_{i \in data} 2(z_3 - y_{3i})
\end{aligned}$$

$$
\begin{aligned}
\frac{\delta E}{\delta w_{(x,1)}} &= \frac{\delta E}{\delta z_2}\frac{\delta z_2}{\delta w_{(1,2)}} + \frac{\delta E}{\delta z_3}\frac{\delta z_3}{\delta w_{(1,3)}} \\
&= \frac{\delta E}{\delta z_2}\frac{\delta z_2}{\delta z_1}\frac{\delta z_1}{\delta w_{(x,1)}} + \frac{\delta E}{\delta z_3}\frac{\delta z_3}{\delta z_1}\frac{\delta z_1}{\delta w_{(x,1)}} \\
&= xw_{(1,2)}h'(z_1 \cdot w_{(1,2)})h'(x \cdot w_{(x,1)}) \sum_{i \in data} 2(z_2 - y_{2i}) \\
&+ xw_{(1,3)}h'(z_1 \cdot w_{(1,3)})h'(x \cdot w_{(x,1)}) \sum_{i \in data} 2(z_3 - y_{3i}) \\
&= xw_{(1,2)}(h(z_1 \cdot w_{(1,2)}) - h(z_1 \cdot w_{(1,2)})^2)(h(x \cdot w_{(x,1)}) - h(x \cdot w_{(x,1)})^2) \sum_{i \in data} 2(z_2 - y_{2i}) \\
&+ xw_{(1,3)}(h(z_1 \cdot w_{(1,3)}) - h(z_1 \cdot w_{(1,3)})^2)(h(x \cdot w_{(x,1)}) - h(x \cdot w_{(x,1)})^2) \sum_{i \in data} 2(z_3 - y_{3i})
\end{aligned}
$$

## 2 Coding Component

```
from PIL import Image
from torch import Tensor, tensor
from torch.utils.data import Dataset, DataLoader
from torchvision.models import resnet18
from torchvision.transforms import *
from getimagenetclasses import parsesynsetwords, parseclasslabel
import sys


class cropSet(Dataset):

    def __init__(self, path, xmlDir, size):
        self.path = path
        self.xmlDir = xmlDir
        self.size = size
        self.transform = None

    def __len__(self):
        return self.size

    def __getitem__(self, idx):
        path = self.path.format(idx + 1)
        im = Image.open(path).convert("RGB")

        idx, name = parseclasslabel(self.xmlDir.format(idx + 1), syn2idx)

        if self.transform:
            im = self.transform(im)
```

```
        return im, idx


def center(img):
    C, W, H = img.size()
    crop = img.new_empty((3, 224, 224))
    x = img.narrow(1, (W - 224) // 2, 224)
    crop = x.narrow(2, (H - 224) // 2, 224)

    return crop


class centerSet(cropSet):

    def __init__(self, *arg):
        super().__init__(*arg)
        self.transform = \
            Compose([
                Resize(224),
                ToTensor(),
                Normalize(mean=[0.485, 0.456, 0.406],
                          std=[0.229, 0.224, 0.225]),
                center,
            ])


def five(img):
    C, W, H = img.size()
    crop5 = img.new_empty((5, 3, 224, 224))
    x = img.narrow(1, 0, 224)
    x = x.narrow(2, 0, 224)
    crop5[0] = x
    x = img.narrow(1, W - 224, 224)
    x = x.narrow(2, 0, 224)
    crop5[1] = x
    x = img.narrow(1, W - 224, 224)
    x = x.narrow(2, H - 224, 224)
    crop5[2] = x
    x = img.narrow(1, 0, 224)
    x = x.narrow(2, H - 224, 224)
    crop5[3] = x
    x = img.narrow(1, (W - 224) // 2, 224)
    x = x.narrow(2, (H - 224) // 2, 224)
    crop5[4] = x
```

```python
    return crop5


class fiveSet(cropSet):

    def __init__(self, *arg):
        super().__init__(*arg)
        self.transform = \
            Compose([
                Resize(280),
                ToTensor(),
                Normalize(mean=[0.485, 0.456, 0.406],
                          std=[0.229, 0.224, 0.225]),
                five,
            ])


def test(model, dataloader):
    total = 0
    correct = 0
    for n, sample_batched in enumerate(dataloader):
        data, descs = sample_batched
        X = data.view(-1, 3, 224, 224)
        out = model.forward(X)
        out = out.view(batch_size, -1, 1000)
        out = out.mean(1)
        val, pred = out.max(1)
        cmp = pred.eq(descs)
        total += cmp.size(0)
        correct += cmp.sum()

    return int(correct), total

def main(dataDir, xmlDir, synDir, n=250, batch_size=10, is_shuffle=False):
    idx2syn, syn2idx, syn2desc = parsesynsetwords(synDir)
    fivecropset = fiveSet(dataDir, xmlDir, n)
    fivecroploader = DataLoader(
        fivecropset, batch_size=batch_size, shuffle=is_shuffle)
    centercropset = centerSet(dataDir, xmlDir, n)
    centercroploader = DataLoader(
        centercropset, batch_size=batch_size, shuffle=is_shuffle)
    model = resnet18(pretrained=True)
    model.eval()
    fiveResult = test(model, fivecroploader)
```

```python
    centerResult = test(model, centercroploader)

    return fiveResult, centerResult


if __name__ == "__main__":
    fiveResult, centerResult = main(sys.argv[1], sys.argv[2], sys.argv[3])
    print("Five Crop Accuracy: %s" % fiveResult)
    print("Center Crop Accuracy: %s" % centerResult)
```